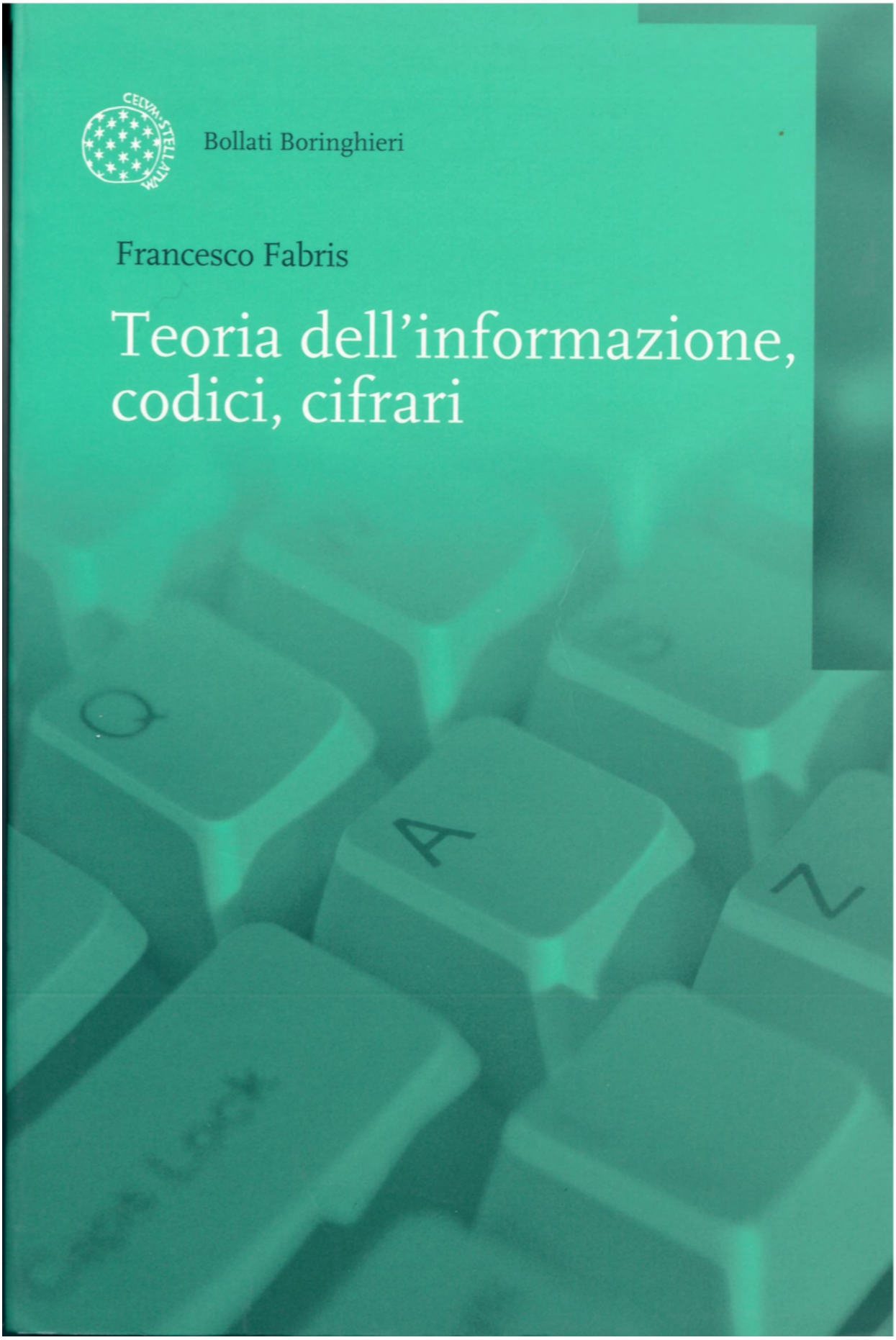

Bollati Boringhieri
Francesco Fabris
Teoria dell'informazione, codici, cifrari

FRANCESCO FABRIS

TEORIA DELL'INFORMAZIONE, CODICI, CIFRARI

A GAGC

Indice

0.1 Prefazione

Questo testo è stato pensato come sussidio didattico per gli studenti delle Lauree di I e II livello relative alle classi di *Scienze e Tecnologie Informatiche* e di *Ingegnerie dell'Informazione*.
Esso è suddiviso in tre parti, le quali curano rispettivamente gli aspetti normativi della *teoria dell'informazione* (o *teoria di Shannon*), lo studio della teoria dei *codici di sorgente* e dei codici *correttori d'errore* e, ultima in ordine d'apparizione, ma non certo in ordine d'importanza, la comprensione dei modelli matematici che sono alla base delle moderne tecniche crittografiche per la protezione (attiva e passiva) dei dati mediante *cifrari*.
La scelta degli argomenti e il livello di approfondimento sono modulati sulla base delle esigenze didattiche complessive che si possono manifestare avendo a disposizione l'equivalente di 12 crediti (circa un centinaio di ore di lezione frontale). La strutturazione del libro in tre *parti* offre quantomeno due percorsi didattici distinti. Il primo può essere basato su tre moduli da 4 crediti (poco più di una trentina d'ore ciascuno), che corrispondono grosso modo alle tre parti in oggetto. In tale ipotesi la prima di esse può considerarsi propedeutica alle altre due, che sono per il resto indipendenti. Il secondo percorso, basato su due moduli da 6 crediti (circa 50 ore), potrebbe essere costituito dalla parte normativa sulla teoria di Shannon arricchita con alcuni esempi di codici di sorgente e di canale, seguita da un secondo modulo di approfondimento sulla teoria dei codici e di studio della crittografia. In entrambi i percorsi sarebbe opportuno sviluppare il primo modulo preferibilmente nel primo triennio della *Laurea*, lasciando la teoria dei codici e dei cifrari al secondo biennio della *Laurea Specialistica* di entrambe le classi.
Non si pensi tuttavia che la trattazione congiunta, in un unico libro, degli argomenti sopra esposti costituisca una forzatura dal punto di vista concettuale; è ben noto che tanto la teoria dell'informazione (e dei codici) che la crittografia devono il loro assetto normativo attuale ai celeberrimi lavori di Claude Elwood Shannon [77, 78], che ebbe il grande merito scientifico di dotare di un modello matematico efficace e stabile tanto il processo di trasmissione dell'informazione quanto quello relativo alla protezione della stessa nei confronti di utenti non autorizzati.

Nella prima parte del libro si forniscono i fondamenti normativi della teoria di Shannon, cioè lo studio sistematico delle misure d'informazione (divergenza, mutua informazione, entropia), dei teoremi asintotici per la codifica di sorgente e di canale, oltre alla descrizione della sorgente d'informazione nei termini delle sequenze tipiche. Nella seconda parte si affrontano alcuni esempi concreti di codici per la compressione dei dati (tanto da lunghezza variabile a blocco che da blocco a lunghezza variabile), ponendo l'accento anche su problemi strettamente legati alle applicazioni (p.es. la codifica adattativa). Si studiano inoltre i codici

(algebrici) correttori d'errore, fornendo una panoramica sufficientemente ampia e articolata del settore. Nella terza parte si descrive il modello della *crittografia a chiave segreta*, legata al contesto storico-normativo della teoria di Shannon, congiuntamente alla nuova fisionomia che la disciplina ha acquisito sotto la spinta derivante dall'introduzione delle *funzioni unidirezionali* (e in genere della *Teoria della Complessità*), che hanno generato la branca della *crittografia a chiave pubblica*.

Le finalità di questo testo sono eminentemente didattiche; di ciò risentono lo stile d'esposizione, la concatenazione e la scelta degli argomenti, il loro livello d'approfondimento. Lo scopo è quello di raccogliere e mettere a disposizione degli interessati, in una trattazione unitaria, una selezione didatticamente sperimentata del materiale riguardante tre discipline vastissime (e tutto sommato ancora poco studiate in Italia), che costringerebbero altrimenti a lunghe ricerche in biblioteche universitarie spesso sprovviste di molti tra i migliori testi o riviste disponibili nella letteratura internazionale (preminentemente di lingua Inglese).

Ciò non deve però esimere il lettore più accorto dall'approfondire, quando le biblioteche lo consentano, lo studio dei testi classici che hanno fatto la storia delle tre discipline; anche a distanza di decine d'anni molti di essi continuano a costituire dei punti di riferimento assoluto dai quali non può prescindere chiunque abbia in mente di leggere (o di scrivere) qualcosa di attendibile in questo campo, tanto in ambito strettamente didattico quanto, molto spesso, anche in ambito più specificamente tecnico-scientifico. Elenchiamo brevemente tali testi fondamentali, sui quali questo stesso libro è basato, riservandoci di specificare meglio, nel corso dei capitoli, i punti che è opportuno approfondire o estendere.

Per quanto riguarda gli aspetti normativi della Teoria dell'Informazione e della crittografia citiamo nuovamente i due lavori fondamentali di Shannon [77, 78], il primo dei quali dà un assetto pressoché definitivo a molte parti della Teoria dell'Informazione; essi sono comunque il punto di partenza di tutte e tre le discipline. Di estremo rilievo per gli aspetti generali sono inoltre i libri (in ordine cronologico) di Ash [3], Gallager [36] e soprattutto di Csizár e Körner [21], manuale molto avanzato e ricco di spunti per la ricerca.

Molto importanti anche il libro di Gray [41] per le generalizzazioni dei teoremi di codifica a processi non stazionari e non ergodici , e quello di Berger [8] per la teoria della distorsione . Ricordiamo inoltre il recente numero speciale della rivista *IEEE Transactions on Information Theory* [44], indubbiamente la più autorevole e specializzata del settore, che ha dedicato un fascicolo commemorativo al 50° anniversario della teoria dell'informazione. In esso troviamo i contributi di eminentissimi Autori, che delineano le direttive di progresso della disciplina nei cinquant'anni trascorsi dai lavori di Shannon, offrendo anche una rassegna dei più recenti sviluppi.

Per la teoria dei codici di sorgente citiamo un testo italiano, quello di Longo [55] (di cui purtroppo non è mai stata curata una traduzione in Inglese), molto completo e con un'introduzione di notevole spessore.

Nell'ambito dei codici correttori la bibliografia è vastissima; per i codici algebrici riferiamo senz'altro il volume di Berlekamp [9], il libro di Peterson e Weldon [66] e, soprattutto, il manuale di MacWilliams e Sloane [59], probabilmente il libro più completo mai scritto su questo tema. Per i *codici convolutivi* (non trattati in questa sede) spiccano invece il libro di Viterbi-Omura [85] e il volume di Piret [67], a essi interamente dedicato.

È inoltre disponibile da poco tempo un'opera straordinaria, curata da Pless e Huffman e rivolta per lo più ai ricercatori del settore, dal titolo *Handbook of Coding Theory* [68]. In due tomi da oltre mille pagine ciascuno, l'opera fornisce una panoramica aggiornatissima su tutta la teoria dei codici correttori, tanto algebrici che convolutivi, offrendo inoltre innumerevoli spunti per la ricerca in questo settore.

Per quanto attiene invece le tecniche crittografiche ricordiamo il testo monumentale di Kahn sulla storia della crittografia [46], oltre ai volumi di Beker e Piper [5], Rüeppel [71], Denning [23] e, molto recente e aggiornato, il testo di Stinson [82]. Fondamentale è l'opera di rassegna *Contemporary Cryptology*, curata da Simmons , che delinea le direttive principali di sviluppo della crittografia secondo una descrizione fatta da alcuni tra i più prestigiosi matematici che hanno contribuito in modo rilevante al progresso della disciplina [80].

Si segnala inoltre il manuale *Handbook of Applied Cryptography* [64], molto completo, ricco e aggiornato, curato da Menezes, van Oorschot e Vanstone, e che offre una rassegna sistematica di tutti i cifrari e protocolli disponibili.

L'esperienza didattica sulla base della quale è stata elaborata la struttura di questo testo è il frutto di un lungo processo di affinamento che riflette, comprende e aggrega anche (e soprattutto) l'esperienza e le qualità didattiche e scientifiche di tutti quei Docenti che nel corso degli ultimi anni hanno insegnato questa disciplina nelle università di Udine e Trieste. A Giacomo Della Riccia, Giuseppe O. Longo, Amelia Giuseppina Nobile e Andrea Sgarro vanno tutta la mia stima e riconoscenza.

Un ringraziamento particolare devo inoltre ad Andrea Sgarro, che con i suoi numerosi e acuti commenti ha migliorato non poco la struttura e la leggibilità di questo testo.

Capitolo 1
Introduzione generale

Come già parzialmente anticipato nella prefazione, teoria dell'informazione, teoria dei codici e crittografia hanno in comune un contesto fisico, quello del *canale* sul quale si effettuano le trasmissioni, e un contesto storico, legato agli straordinari articoli di Shannon precedentemente citati. La crittografia ha in realtà una propria storia antichissima: la prima trasformazione di un testo in senso crittografico di cui si abbia notizia sembra risalire ad alcuni geroglifici egiziani (si veda il volume di Kahn [46]), ma solo grazie al fertile lavoro di Shannon essa ha acquisito uno status di moderna disciplina scientifica, associata a un modello matematico completo e formalizzato in modo adeguato.

Viceversa la teoria dell'informazione nacque solo nel momento in cui Shannon riuscì a delinearne i principali aspetti normativi, legati in prima approssimazione alla definizione dell'*entropia* (di sorgente) come valor medio dell'*autoinformazione*, e della *capacità* (di canale) come valor massimo della *mutua informazione* (per una impostazione diversa si veda [47, 48]). Non ci dilungheremo molto in una riflessione storico-filosofica sul significato della parola *informazione*, sul contesto in cui queste prime ricerche si svilupparono e sull'impatto che le teorie di Shannon hanno avuto in tutti i molteplici aspetti della scienza dell'elaborazione dell'informazione in generale e dell'Informatica in particolare. Ciò è dovuto alla circostanza che difficilmente si potrebbe discernere su tali argomenti meglio di quanto sia stato fatto in numerosi contributi di G.O. Longo [55, 56, 57] (di cui ricordiamo sopra tutti "Nuovo Golem" e "Homo technologicus") , ed è a questi scritti che rimandiamo il lettore interessato ad approfondire le molteplici sfaccettature relative alla genesi e alla definizione operativa dell'informazione, e soprattutto alle modificazioni strutturali che la fitta rete di canali di comunicazione a basso costo oggi disponibili sta portando in seno alla nostra società tecnologica; si pensi a tal proposito che i sociologi usano oramai la locuzione *società dell'informazione* come la più pertinente a riassumere l'impatto che le nuove tecnologie informatiche e multimediali hanno sulla vita di tutti i giorni.

La Teoria dell'Informazione in generale e lo studio dei Codici in particolare si svilupparono storicamente come risposta a problemi essenzialmente ingegneristici, dell'ambito delle telecomunicazioni, anche se di queste discipline furono riconosciute ben presto le potenzialità significative anche dal punto di vista matematico. Si pensi per esempio ai legami tra divergenza informazionale e *Statistica* [53], tra entropia shannoniana e *Termodinamica* [19], tra entropia e *Complessità algoritmica di Kolmogorov* [83, 51, 52, 16] o ancora tra teoria dell'informazione e *Teoria estremale dei grafi*). Non meno proficua risulta l'interazione tra *Algebra* (campi di Galois e Teoria dei Gruppi), *Geometria* (sistemi di Steiner e curve algebriche) e codici correttori, o ancora tra *Teoria dei Numeri*, *Teoria della Complessità* e Crittografia. Molto spesso il ruolo di queste discipline matematiche è stato quello di fornire strumenti di straordinaria efficacia per la soluzione di problemi inerenti la teoria delle comunicazioni e l'informatica; in qualche caso, però, i problemi della Teoria dell'Informazione in particolare e dell'Informatica più in generale sono stati fonte d'ispirazione per i matematici, suggerendo loro nuove direttive di ricerca in un meccanismo proficuo di retroazione che ha portato molti grandi matematici del nostro secolo ad occuparsi *anche* di queste discipline.

In questa sede, oltre a dar ampio spazio alla parte normativa e a quella più propriamente legata alle applicazioni, rifletteremo anche sulle motivazioni logico-strutturali che hanno portato a riconoscere l'entropia di Shannon e la capacità di canale quali interpreti principali sulla scena del sistema di comunicazione, dandone ampia giustificazione sul piano matematico-formale.

Il nostro percorso didattico non ha dunque come punto di partenza una prospettiva tecnico-storica, come di solito avviene nella quasi totalità dei testi di Teoria dell'Informazione, quanto piuttosto una *ipotetico-deduttiva*, che fa riferimento prioritariamente alla *divergenza informazionale* (o *divergenza di Kullback*), dalla quale derivano tutte le altre *misure d'informazione* significative in questo contesto. Questo è il motivo per cui la trattazione delle misure d'informazione, solitamente relegate nei testi specialistici a un ruolo di mero strumento operativo (e comunque subordinato alle esigenze relative al contesto delle telecomunicazioni), verrà in questa sede posta come punto di partenza, occupando il primo capitolo.

Questa impostazione può risultare utile anche a quegli studenti delle classi di *Scienze Matematiche* che abbiano interesse ad approfondire questi aspetti della matematica applicata, senza dover necessariamente legarsi, nello studio, a giustificazioni o motivazioni ingegneristico-operative, del tutto avulse dal contesto in cui si opera in un corso di laurea di Matematica.

In ogni caso, prima di affrontare le *misure d'informazione*, è opportuno fornire al lettore una breve descrizione intuitiva del modello fisico dal quale si parte, e che è stato il punto di riferimento costante nello sviluppo di tutte e tre le discipline.

1.1 Modello di un sistema di trasmissione

1.1.1 Informazione

Lo scopo fondamentale della Teoria dell'Informazione (nella sua accezione più ampia comprendente anche lo studio dei Codici e dei Cifrari) è quello di trasferire *l'informazione* a *distanza* in modo affidabile, preciso, economico e sicuro. Nel modello shannoniano l'informazione viene generata da una *sorgente d'informazione* $\boldsymbol{S}$ e la distanza che questa deve percorrere può avere tanto natura *spaziale* che *temporale*. Nel primo caso l'informazione giunge, attraverso un opportuno *canale*, a un *utente* $\boldsymbol{U}$ che la riceve, e che ha una collocazione spaziale diversa da $\boldsymbol{S}$; nel caso di distanza temporale si può pensare che l'informazione venga *memorizzata* e resa diponibile solo in un secondo momento (si pensi al caso delle memorie di massa degli elaboratori).

L'informazione non ha una valenza assoluta, ma dipende in sostanza dalla capacità di discriminazione dell'osservatore a essa interessato; questi ha di fronte a sé un sistema fisico, la *sorgente*, e una (o più) grandezze che lo descrivono, suscettibili di assumere *più stati diversi*. L'informazione nasce, come stato oggettivo per l'osservatore, nel momento in cui questi individua una *differenza* nella grandezza di riferimento in corrispondenza di due istanti successivi. Inoltre la quantità d'informazione fornita in un esperimento è intuitivamente legata alla numerosità degli stati possibili *a priori*.

L'informazione rilevata da un secondo osservatore, più acuto del primo e che sia in grado di discriminare tra più stati diversi della stessa grandezza (o tra più grandezze), rimane per il primo osservatore in uno stato di latenza, che si desta solo nel momento in cui anche il primo osservatore dispone della capacità fisico-sensoriale (e della volontà) di rilevare le differenze di cui prima era inconsapevole. Come esempio si può pensare all'esperimento "lancio di un dado", con un osservatore O_1 (il "giocatore") interessato solamente al numero uscito, e un osservatore O_2 (lo studente di Fisica) che apprezza anche l'orientamento delle facce, la posizione occupata dal dado nell'istante finale ecc. I due punti di vista sull'informazione associata all'esperimento sono molto diversi, ma potrebbero coincidere nel momento in cui il "giocatore" decidesse d'iscriversi a un corso di Fisica o, viceversa, lo studente di Fisica decidesse di dedicarsi al gioco d'azzardo, ritenendo di aver compreso, a seguito degli esperimenti, quale informazione sia *effettivamente* utile per i propri fini (ma per il concetto di *informazione utile* si veda anche la sezione 2.4).

Le *differenze* di cui si parla hanno una loro strutturazione gerarchica, poiché può *cambiare* il modo in cui si manifesta una differenza; anche questa differenza del secondo livello (differenza di differenze) è rilevabile in modo oggettivo ed è essa stessa fonte d'informazione (si veda a tal proposito l'eccellente introduzione del testo di Longo [55]). Riprendendo l'esempio del dado si può pensare

a una successione di lanci, fatta dallo studente di Fisica, in cui ciascuna faccia abbia una frequenza relativa di (circa) 1/6, seguita da una seconda successione "sospetta", eseguita dal giocatore d'azzardo, in cui il 6 esca con una frequenza stranamente elevata. Se dal punto di vista delle differenze del prim'ordine ci si può limitare a registrare l'accaduto, un'analisi dell'informazione associata alle differenze del second'ordine, cioè al diverso "comportamento" del dado lanciato dal giocatore, porterebbe a un'informazione di secondo livello su una presunta manomissione del dado da parte del giocatore.

Questa gerarchia si colloca comunque su un piano strettamente *sintattico* (nell'esempio precedente l'informazione è relativa al solo fatto che il comportamento del dado è cambiato). Tuttavia *l'interpretazione* delle differenze relativamente a un certo contesto logico, l'unico che possa esprimere un *significato* per esse (la conseguente congettura sul dado truccato), induce una loro stratificazione sul piano *semantico*, molto difficile da gestire con un modello matematico formale.

L'informazione ha dunque un carattere relativo, dipende sensibilmente dall'utente che interroga la sorgente, dalla sua capacità di discriminazione, e presuppone una fase sintattica o di *rilevamento* della stessa che si distingue nettamente dalla fase semantica relativa alla *comprensione* del significato, processo che consiste essenzialmente nel riportare ad una propria esperienza pregressa i caratteri sintattici emersi dall'informazione disponibile, riconducendo la massa disaggregata di dati verso piani di coerenza semantica.

Come esempio illuminante si rifletta sulla lettura di una frase scritta in una lingua nota, contrapposta alla lettura di una frase scritta in una lingua ignota. In entrambi i casi avviene il rilevamento, ma la comprensione del significato della frase si ha solamente nel primo caso.

Ciò ha come immediata conseguenza un'intrinseca difficoltà nel giungere a una quantificazione globale dell'informazione, che tenga cioè conto tanto dei suoi aspetti sintattici quanto di quelli semantici, che possono fra l'altro essere assai poco correlati tra loro.

Il modello suggerito da Shannon fa riferimento al solo piano sintattico, e ciò costituisce contemporaneamente un pregio e una limitazione. La limitazione è conseguente alla circostanza che, essendo svincolato da connotazioni di tipo semantico, il modello descrive solo in minima parte la ricchezza del processo di comunicazione, così come avviene nel mondo reale. Il pregio risiede nel fatto che tale limitazione, circostanziando la validità del modello, ha consentito di ottenere risultati rilevanti sul piano tecnico-operativo (i vari teoremi di codifica), che hanno dato un impulso straordinario allo sviluppo di queste discipline.

I soggetti principali del *sistema di comunicazione unidirezionale* di figura 1.1 sono dunque, in prima analisi, la *sorgente* $\boldsymbol{S}$ che emette l'informazione, il *canale* $\boldsymbol{C}$ attraverso il quale questa transita e *l'utente legittimo* $\boldsymbol{U}$, l'unico ad avere

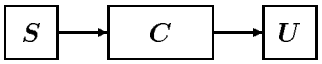


Figura 1.1 Schema di un semplice sistema di comunicazione unidirezionale.

titolo di accesso all'informazione. Il canale C è il mezzo fisico attraverso il quale le differenze di cui prima si parlava si propagano al di fuori della sorgente. In generale il canale potrebbe essere un qualunque sostrato fisico che consenta la lettura a distanza della configurazione assunta dalla sorgente, ma in pratica, perlomeno nel contesto tecnico-ingegneristico in cui di fatto si finisce con l'operare, esso è quasi sempre una linea di trasmissione (doppino telefonico, cavo coassiale, guida d'onda o fibra ottica), scelta sulla base della tecnologia in auge, e che consenta la trasmissione a distanza dei cosiddetti *segnali* che incorporano l'informazione che la sorgente emette. Le sorgenti e i canali, e dunque i segnali che in essi transitano, possono avere tanto natura *continua* che *discreta* (sistemi *analogici* o sistemi *numerici*) , anche se il celebre *teorema del campionamento*, risultato centrale nella teoria delle comunicazioni elettriche, stabilendo la possibilità di approssimare con errore asintoticamente nullo un segnale analogico con uno discreto, ha decretato di fatto l'abbandono della tecnologia analogica. Inoltre, all'interno del dominio del discreto si è assistito a uno sviluppo straordinario dei sistemi *binari*, derivante in ultima analisi da fattori strettamente tecnici, anche se corroborati dallo sviluppo maturo dell'*algebra di Boole* ad essi associata.

1.1.2 Ridondanza e codifica di sorgente

Queste considerazioni fanno comprendere come sia necessario tradurre l'informazione che esce dalla sorgente, basata su un alfabeto che chiamiamo *primario* o di sorgente, in altra informazione relativa a un alfabeto *secondario* (quello del canale), che molto frequentemente è binario. Tale traduzione va fatta però nel modo più economico possibile, cioè rendendo minime in qualche senso le lunghezze delle sequenze secondarie che entrano nel canale; ciò poiché l'occupazione del canale, o dello spazio di memoria riservato ai messaggi della sorgente, ha un costo che si può considerare proporzionale alla lunghezza della stessa sequenza.

Def. 1.1. *Si definisce codifica di sorgente la traduzione di una sequenza da un alfabeto primario a un alfabeto secondario che soddisfi opportuni criteri di economicità. Essa è attuata da un dispositivo specifico, chiamato codificatore di sorgente (CS), che effettua una compressione dei dati.*

La compressione dei dati elimina l'informazione superflua generata dalla sorgente, la cosiddetta *ridondanza*, cioè quell'informazione non strettamente necessaria per recuperare la piena integrità del messaggio trasmesso. Il codificatore di sorgente attua dunque una *rimozione* della ridondanza, rendendo le sequenze estremamente fragili nei confronti di eventuali alterazioni involontarie delle stesse. In ricezione si dovrà ricorrere a un dispositivo simmetrico, il *decodificatore di sorgente* (DS), che effettui l'operazione inversa di ricostruzione delle sequenze primarie a partire dalle secondarie, rendendo così intelligibili all'utente i dati ricevuti.

Anche la ridondanza ha una duplice natura, sintattica e semantica, che si può comprendere sulla base del seguente esempio: si voglia ricostruire la parte mancante della frase "Il primo mese dell'anno è Ge . . . ", basandosi prima sulla frequenza relativa delle lettere in Italiano (o delle coppie di lettere, terne ecc, che è un criterio di tipo sintattico), e poi sul significato in Italiano della frase. Nel primo caso una ricostruzione plausibile potrebbe essere "Ge*rosa*", sintatticamente verosimile, mentre nel secondo caso si giunge senza dubbio alla parola corretta Ge*nnaio*. In altri termini la presenza di ridondanza in una sequenza consente di ricostruire in modo corretto l'informazione mancante.

1.1.3 Rumore e codifica di canale

La sequenza secondaria che esce dal codificatore di sorgente non può essere trasmessa direttamente attraverso il canale, poiché molto probabilmente ne uscirebbe deturpata dal *rumore*, sempre presente in tutti i sistemi fisici di trasmissione dei segnali. Non dimentichiamo infatti che la sequenza è stata sfrondata dalla ridondanza, ed è quindi particolarmente vulnerabile. Il rumore è legato in prima approssimazione all'agitazione termica dei materiali, sempre presente a meno che non si operi a 0 gradi Kelvin. Ma ci sono anche altre componenti, che possono assumere un valore significativo soprattutto in corrispondenza di determinate frequenze (p.es. il rumore "flick"). Senza entrare nel dettaglio della tipologia di rumore associata alle diverse linee di trasmissione, l'unica cosa che interessa in questa sede è che, qualunque sia il sistema usato, al segnale che porta l'informazione si somma un certo *segnale di rumore*.

Nel caso dei sistemi puramente analogici tale sovrapposizione risulta essenzialmente irreversibile, nel senso che l'unico stratagemma da impiegare dal punto di vista tecnico per contenere l'effetto del rumore è in ultima analisi quello di usare dei segnali di ampiezza elevata, portando così a un aumento del cosiddetto *rapporto segnale-rumore*.

Viceversa, nel caso in cui il segnale analogico venga impiegato come sostegno (o *portante*, come si dice in gergo ingegneristico) di un segnale *discreto*, c'è la possibilità di recuperare perfettamente l'integrità dell'informazione. Se

infatti il rumore non è stato "troppo" forte si possono separare senza incertezza i livelli del segnale che corrispondono ai diversi stati logici; nel caso binario si tratta per esempio di distinguere la presenza del segnale portante, che corrisponde p.es. a "1", dalla sua assenza, cioè "0", cosa relativamente semplice anche nel caso in cui entrambi i livelli degli stati logici siano sovrapposti a un (debole) segnale di rumore.

Per inciso osserviamo che anche questa minore insofferenza nei confronti del rumore da parte dei sistemi discreti ha avuto un ruolo importante nel superamento della tecnologia analogica a favore di quella numerica (o *digitale*). Altro elemento che ricordiamo brevemente è una maggior *affidabilità* dei circuiti di tipo numerico, associata alle diverse condizioni di lavoro che sono generalmente meno gravose di quelle che si realizzano con segnali di tipo analogico.

Nel caso discreto l'effetto del rumore sull'informazione è dunque quello di una possibile alterazione degli stati logici (nel caso binario 0 può diventare 1 o viceversa); il modello che conviene adottare è allora di tipo *probabilistico*, introducendo le cosiddette *probabilità d'errore* P_{01} e P_{10} (ma molto spesso $P_{01} = P_{10} = \epsilon$, con ϵ che dipende fortemente dal tipo di linea di trasmissione). Anche se la probabilità d'errore è numericamente molto piccola, una sequenza secondaria sufficientemente lunga conterrà prima o poi almeno un errore.

Il metodo per proteggersi dal rumore è quello d'inserire in modo sistematico nelle sequenze secondarie della ridondanza *strutturata*, che consenta di recuperare quella parte della sequenza che dovesse risultare inquinata dal rumore. Ciò comporta evidentemente un'espansione dei dati, finalizzata però alla loro protezione.

Def. 1.2. *Si definisce codifica di canale l'inserimento deliberato di ridondanza strutturata in una sequenza, finalizzata alla protezione della stessa dal rumore. L'operazione viene effettuata da un dispositivo chiamato codificatore di canale (CC), che viene posto tra codificatore di sorgente CS e canale C.*

In ricezione dobbiamo disporre di un *decodificatore di canale* (DC), che elimini la ridondanza strutturata e che individui (e possibilmente corregga) gli errori intercorsi per effetto del rumore. Se tutto funziona la sequenza che si ha a disposizione corrisponde a quella in uscita al codificatore di sorgente CS; la decodifica della sequenza mediante il successivo decodificatore di sorgente (DS) rende nuovamente intelligibile l'informazione all'utente legittimo (si veda la figura 1.2). Codifica di sorgente e codifica di canale sono di stretta pertinenza della Teoria dell'Informazione, che deve individuare i *codici* più idonei per effettuare la compressione dei dati e la protezione degli stessi dal rumore.

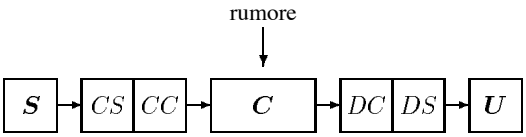


Figura 1.2 Codifica di sorgente e di canale in un sistema di comunicazione unidirezionale.

1.1.4 Segretezza e cifratura dei messaggi

Il rumore non è però l'unico ospite indesiderato del modello di trasmissione di figura 1.2. Se i dati che transitano sul canale hanno una natura *riservata*, la loro segretezza deve essere salvaguardata nei confronti di un possibile *utente non autorizzato* in agguato sul canale e desideroso di acquisire le informazioni riservate. La protezione dei dati non può tuttavia essere affidata alla speranza di una mancata rilevazione degli stessi da parte dell'intruso, poiché si deve ragionevolmente presupporre che egli sia in possesso della tecnologia necessaria per "leggere" tutte le sequenze che transitano sul canale. La salvaguardia sul piano sintattico può però avvenire rendendo il canale *inaccessibile* o indisponibile alla spia. Per tutta una serie di motivi che analizzeremo nel seguito questa soluzione è praticabile solo in linea di principio, per canali particolari o relativamente a messaggi di lunghezza modesta.

Bisogna dunque provvedere a una protezione che tocchi il piano semantico. Un metodo ovvio è quello di effettuare delle trasformazioni sul messaggio che bisogna trasmettere, detto anche *messaggio in chiaro*, in modo da ottenere il cosiddetto *messaggio cifrato*. Se la trasformazione rimane ignota alla spia, l'eventuale rilevamento del testo cifrato, o *crittogramma*, da parte di quest'ultima non consente comunque la comprensione del messaggio in chiaro.

Def. 1.3. *L'operazione di trasformazione da testo in chiaro a testo cifrato si chiama cifratura, e viene attuata da un sistema cifrante (SC) che può essere posto tra sorgente e codificatore di sorgente CS.*

L'operazione inversa alla cifratura, che consente di riottenere il messaggio in chiaro conoscendo la trasformazione usata (o *chiave* di cifratura), si chiama *decifrazione*, e si avvale di un opportuno *sistema decifrante* (SD).

Si osservi che la decifrazione, effettuata dall'utente legittimo, viene solitamente distinta dalla *decrittazione*, che è l'operazione *illegittima* di recupero del messaggio in chiaro senza avvalersi della conoscenza della chiave. Dal punto di vista operativo decifrazione e decrittazione sono strutturalmente diverse, poiché hanno un diverso livello di *complessità computazionale*. È infatti evidente che la decifrazione dev'essere un'operazione *facile*, mentre la decrittazione dev'essere computazionalmente *complessa* (possibilmente "intrattabile").

1.1.5 Frodi e autenticazione dei messaggi

L'utente non autorizzato, acquattato in prossimità del canale e dedito a un'attività *passiva* di acquisizione dei dati (passiva dal punto di vista del processo di comunicazione sorgente-utente legittimo), può tentare anche un'intrusione di tipo *attivo*, *impersonando* la sorgente che sta trasmettendo e tentando di far attribuire a quest'ultima dei messaggi *falsi*, fraudolentemente inseriti da egli stesso nel sistema di comunicazione (di solito in prossimità del canale, che è il blocco funzionale più esposto agli attacchi esterni). Tale frode è stata storicamente combattuta, nel caso di documenti cartacei, con la *firma* in calce al documento, che ha la doppia funzione di impedire le impersonazioni da parte di falsari e di impedire al mittente la ripudiabilità di un documento (o messaggio) regolarmente trasmesso.

Una tale situazione deve poter essere estesa anche al contesto dei sistemi di comunicazione non più basati su supporto cartaceo, introducendo una sorta di *firma numerica* che autentichi l'identità della sorgente o del documento che si sta trasmettendo. Per ottenere l'autenticazione il mittente deve *dimostrare* all'utente legittimo di essere l'unico a saper effettuare una certa operazione, per esempio generare un crittogramma la cui decifrazione porti a un breve messaggio col nome proprio del mittente (più eventualmente altre informazioni di carattere contestuale e temporale per evitare che la stessa firma possa essere usata in un momento successivo).

Def. 1.4. *L'insieme delle procedure necessarie per evitare le frodi da impersonazione si chiama autenticazione della sorgente; essa viene effettuata da un sistema di autenticazione SA che fornisce una firma numerica identificativa della sorgente legittima.*

L'autenticazione della firma viene effettuata in ricezione da un dispositivo, che chiamiamo *sistema di identificazione* (SI), che dovrà riconoscere la legittimità della sorgente.

Accenniamo infine al fatto che l'attacco attivo al sistema di comunicazione può avere una connotazione più subdola, che non coinvolge direttamente l'identità della sorgente quanto piuttosto *l'integrità* dei dati trasmessi dalla sorgente legittima. Si pensi al caso di un ordine di trasferimento di danaro, p.es. Lit 100.000, eseguito dalla banca autorizzata $\mathfrak{B}$ a favore dell'utente signor *Hacker*, nell'ipotesi in cui detto utente sia appostato in prossimità del canale nel momento in cui transita l'ordine; la tentazione del signor *Hacker* di aggiungere qualche zero al credito in transito potrebbe portare ad una deliberata *alterazione* dei dati trasmessi dalla sorgente legittima $\mathfrak{B}$, che compromette *l'integrità* degli stessi.

La progettazione degli algoritmi di cifratura, di autenticazione e di verifica dell'integrità dei dati, cioè dei *cifrari* in senso lato, costituisce l'oggetto

di studio della *crittografia*. L'analisi dei metodi più idonei che un oppositore potrebbe escogitare per effettuare un'eventuale *forzatura* degli stessi cifrari è invece di dominio della *crittanalisi*. Crittografia e crittanalisi costituiscono una disciplina unitaria denominata *Crittologia*. In questo testo ci occuperemo più frequentemente della fase creativa, cioè della crittografia. Nel seguito forniremo le indicazioni bibliografiche necessarie per approfondire anche gli aspetti fondamentali della crittanalisi.

Lo schema completo di un sistema di comunicazione unidirezionale, nel quale venga attuata una compressione dei dati, una protezione dal rumore, dalle intercettazioni o da attacchi di impersonazione è rappresentato in figura 1.3.

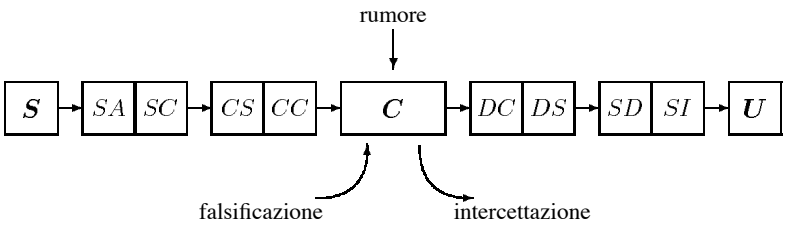


Figura 1.3 Sistema di comunicazione unidirezionale protetto dal rumore, dalle intercettazioni e dalle impersonazioni.

Parte I

Teoria dell'informazione

Capitolo 2
Misure d'informazione

La teoria dell'informazione è legata in modo stretto alla *teoria delle probabilità e delle variabili aleatorie*, nel senso che quest'ultima disciplina offre gli strumenti necessari (e a quanto pare anche sufficienti, visti i grossi successi conseguiti nell'ambito delle telecomunicazioni) a descrivere in modo appropriato il funzionamento del *sistema di comunicazione*. In questo primo capitolo cercheremo d'individuare e studiare le funzioni che sono alla base di tutte le definizioni e i teoremi che prenderanno corpo nell'ambito del modello descritto da Shannon.

2.1 Divergenza informazionale

Nel corso di tutto il libro avremo a che fare con delle *distribuzioni di probabilità* (d.p.) ([34]), cioè funzioni non negative $p(x), x \in \mathcal{X}$ e $\sum_{x\in\mathcal{X}} p(x) = 1$, dove $\mathcal{X} = \{x_1, x_2, \ldots, x_K\}$ è un insieme finito e discreto di elementi che corrisponde solitamente a un *alfabeto*. Se $X, Y, Z, \ldots$ sono variabili aleatorie (v.a.), la *distribuzione di una variabile aleatoria* X è la d.p. P_X definita da $P_X = \{p_X(x), x \in \mathcal{X}\}$, dove $p_X(x) = \Pr\{X = x\}$ (probabilità che la v.a. X assuma il valore x). In modo analogo si definisce la *distribuzione congiunta* P_{XY} delle v.a. X e Y: $P_{XY} = \{p_{XY}(x, y), x \in \mathcal{X}, y \in \mathcal{Y}\}$, con $p_{XY}(x, y) = \Pr\{X = x, Y = y\}$ e dove la v.a. congiunta (X, Y) assume i propri valori nello spazio prodotto $\mathcal{X} \times \mathcal{Y}$. Come noto si ha $\sum_{x,y} p_{XY}(x, y) = 1$ e $\sum_y p_{XY}(x, y) = p_X(x)$. In pratica ci affrancheremo dall'uso dei pedici, usando la notazione $p(x)$ e $p(x, y)$ in vece di $p_X(x)$ e $p_{XY}(x, y)$ (almeno laddove ciò non costituisca fonte di equivoco).

Le varie funzioni che useremo nel seguito, e che saranno interpretate con riferimento al contesto informazionale (entropia, mutua informazione, capacità), dal punto di vista matematico devono la loro coerenza alle proprietà analitiche della funzione *logaritmo*, in particolare alla sua *convessità* $\cap$. Anticipiamo che è possibile *dimostrare* che il logaritmo, legato nel nostro contesto alla definizione

dell'entropia, è l'*unica* funzione che soddisfa un insieme di postulati, i quali delineano le proprietà formali necessarie a una *misura d'informazione*. Partiremo dunque dallo studio della convessità del logaritmo.

È ben noto che la convessità $\cup$ di una funzione f in un intervallo (a, b) resta definita nel caso in cui, per ogni coppia x_1, x_2 di punti appartenenti all'intervallo e per $0 \leq \lambda \leq 1$, si abbia $\lambda f(x_1) + (1-\lambda)f(x_2) \geq f(\lambda x_1 + (1-\lambda)x_2)$. Assegnata una certa v.a. X, e indicata con $E[X]$ la sua *speranza matematica*, la convessità per la f implica la validità della seguente disuguaglianza

Teorema 2.1 (Disuguaglianza di Jensen). *Se f è una funzione convessa $\cup$ vale la seguente disuguaglianza*

$$E[f(X)] \geq f(E[X]) \tag{2.1}$$

Dim. Si deve dimostrare che

$$\sum_{i=1}^{K} p_i f(x_i) \geq f\left(\sum_{i=1}^{K} p_i x_i\right)$$

La dimostrazione procede per induzione. Siano $p_1, p_2, \ldots p_K, 0 \leq p_i \leq 1$ coefficienti reali tali che $\sum_i p_i = 1$. Quando $K = 2$ si ha, per l'ipotesi di convessità, $p_1 f(x_1) + p_2 f(x_2) \geq f(p_1 x_1 + p_2 x_2)$. Supponiamo che l'asserto sia valido per $K-1$ e dimostriamolo per K. Si ponga $q_i = \frac{p_i}{1-p_K}$ $(1 \leq i \leq K-1)$

$$\begin{aligned}\sum_{i=1}^{K} p_i f(x_i) &= p_K f(x_K) + (1-p_K) \sum_{i=1}^{K-1} \frac{p_i}{1-p_K} f(x_i) \geq \\ &\geq p_K f(x_K) + (1-p_K) f\left(\sum_{i=1}^{K-1} q_i x_i\right) \geq \\ &\geq f\left(p_K x_K + (1-p_K) \sum_{i=1}^{K-1} q_i x_i\right) = f\left(\sum_{i=1}^{K} p_i x_i\right)\end{aligned}$$

dove la prima disuguaglianza segue dall'ipotesi induttiva e l'ultima dalla convessità della funzione $\square$

La disuguaglianza di Jensen, applicata in modo opportuno alla funzione logaritmo, consente di ricavare la *disuguaglianza della somma logaritmica*, che prelude alla definizione della *divergenza informazionale*; quest'ultima, come vedremo, ha un ruolo centrale in teoria dell'informazione.

Teorema 2.2 (disuguaglianza della somma logaritmica). *Se $a_1, a_2, \ldots, a_K$ e $b_1, b_2, \ldots, b_K$ sono reali non negativi, allora*

$$\sum_{i=1}^{K} a_i \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{K} a_i\right) \log \frac{\left(\sum_{i=1}^{K} a_i\right)}{\left(\sum_{i=1}^{K} b_i\right)} \tag{2.2}$$

e l'eguaglianza vale se e solo se $a_i/b_i = cost$.

Dim. La funzione $f(x) = x \log x$ è convessa $\cup$, come si può verificare facilmente, e per essa vale la disuguaglianza di Jensen. Posto allora $x_i = a_i/b_i$ e $p_i = b_i / \sum_{j=1}^{K} b_j$ si ha $\sum_{i=1}^{K} p_i = 1$; possiamo dunque scrivere $\sum_{i=1}^{K} p_i f(x_i) \geq f(\sum_{i=1}^{K} p_i x_i)$ cioè

$$\sum_{i=1}^{K} \frac{b_i}{\left(\sum_j b_j\right)} \frac{a_i}{b_i} \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{K} \frac{b_i}{\left(\sum_j b_j\right)} \frac{a_i}{b_i} \right) \log \left(\sum_{i=1}^{K} \frac{b_i}{\left(\sum_j b_j\right)} \frac{a_i}{b_i} \right)$$

che è la tesi $\square$

Si noti che per la funzione $f(x) = x \log x$ definiamo $f(0) = 0$ per continuità, giacché $\lim_{x \to 0} x \log x = 0$ (anche termini come $0 \log(0/0)$ vengono uguagliati a zero). Nel seguito non ci preoccuperemo della base dei logaritmi, che sarà in pratica quasi sempre quella binaria; se così non fosse resta inteso che tutto può essere ricalibrato mediante le opportune costanti che portano da una base all'altra.

Nel caso in cui le a_i e le b_i sommino allo stesso valore, per esempio siano entrambe delle distribuzioni di probabilità, il termine di destra della (2.2) diventa zero, e dalla relazione si può dedurre la seguente

Def. 2.1. *Se $P = \{p_1, p_2, \ldots, p_K\}$ e $Q = \{q_1, q_2, \ldots, q_K\}$ sono due distribuzioni di probabilità, si definisce divergenza informazionale (o di Kullback-Leibler) la quantità*

$$D(P//Q) \triangleq \sum_{i=1}^{K} p_i \log \frac{p_i}{q_i} \tag{2.3}$$

($p_i > 0, q_i \geq 0$). Si verifica subito che la divergenza informazionale, oltre che essere una quantità non negativa a norma della (2.2), vale 0 se e solo se $P \equiv Q$. Inoltre essa non è limitabile superiormente; infatti è sufficiente che anche per una sola coordinata i si abbia $q_i \to 0$ e si ottiene $D(P//Q) \to +\infty$. Tutto ciò può essere esplicitato nell'espressione

$$\begin{aligned} 0 \leq D(P//Q) \leq +\infty \qquad & (= 0 \quad \text{sse} \quad P \equiv Q) \\ & (= +\infty \quad \text{sse} \quad \exists\, i : q_i = 0) \end{aligned} \tag{2.4}$$

che riassume le proprità della divergenza. Si noti che la divergenza è somma di termini positivi e negativi; il fatto che la media sia sempre maggiore (o uguale) a zero non è dunque scontato, ma deriva solo dall'uguaglianza $\sum_i a_i = \sum_i b_i$ nella (2.2). La circostanza che sia $D(P//Q) = 0$ se e solo se $P \equiv Q$ consente d'interpretare in modo naturale la divergenza tra due d.p. come una *pseudo-distanza* tra le stesse. Per essere una vera e propria distanza mancano infatti

tanto la simmetria (in quanto in generale $D(P//Q) \neq D(Q//P)$) quanto la validità della disuguaglianza triangolare (se R è una terza d.p., non è detto che $D(P//R) + D(R//Q)$ prevalga su $D(P//Q)$). Abbiamo già anticipato la centralità dalla divergenza di Kullback nell'ambito della teoria dell'informazione, e di ciò renderemo ampiamente conto nelle prossime sezioni. In questa sede ricordiamo ancora che la divergenza possiede interpretazioni anche come perdita asintotica di ottimalità quando si impieghi un codice disaccoppiato dalla sorgente (sez. 5.5), in senso geometrico nell'ottimalità della codifica di Huffman (sez. 5.2), nella valutazione della P-probabilità di una sequenza $\boldsymbol{x}$ appartenente a un tipo F (par. 3.3.2) e nell'espressione asintotica del rapporto di verosimiglianza nell'ambito della discriminazione tra ipotesi statistiche ([53]) (solo per citare quelle più importanti).

2.2 Mutua informazione ed entropia

Se P_X e P_Y sono due d.p. associate a due v.a. X e Y, l'indipendenza tra le stesse viene stabilita dalla validità della $p_{XY}(x,y) = p_X(x)p_Y(y)$ per ogni coppia $x \in \mathcal{X}, y \in \mathcal{Y}$. Un altro modo per esprimere l'indipendenza è quello di interpretarla come l'unica condizione grazie alla quale, dalla conoscenza delle d.p. marginali P_X e P_Y, si riesce a ricavare in modo univoco la d.p. congiunta P_{XY}. Assegnata una d.p. congiunta P_{XY} può essere allora interessante tentare una valutazione della *distanza dalla condizione di indipendenza* tra le variabili. Facendo uso della divergenza informazionale si tratta di calcolare la $D(P_{XY}//P_XP_Y)$; se c'è indipendenza allora $P_{XY} \equiv P_XP_Y$, e la divergenza è nulla; viceversa, quando esiste una relazione di dipendenza tra X e Y questa può essere *misurata* dalla

$$D(P_{XY}//P_XP_Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \triangleq I(X \wedge Y) \tag{2.5}$$

La quantità $I(X \wedge Y)$ che rimane definita si chiama *mutua informazione* tra le v.a. X e Y. Essa è simmetrica nelle sue variabili ($I(X \wedge Y) = I(Y \wedge X)$), ed è sempre non negativa, in quanto trattasi di una divergenza informazionale. Essendo nulla se e solo se la d.p. congiunta coincide col prodotto delle marginali, cioè quando sussiste l'indipendenza tra X e Y, si ricava che la mutua informazione può essere interpretata come *misura di dipendenza stocastica* tra v.a. Si può evidenziare un'interpretazione più aderente all'ambito informazionale non appena si pensi che il caso d'indipendenza può essere visto, dal punto di vista intuitivo, come una situazione in cui X e Y *non si scambiano* informazione. Di conseguenza la mutua informazione verrà interpretata anche come misura della *quantità media d'informazione* scambiata tra le v.a. X e Y (ma su questo ritorneremo più diffusamente nella sezione 2.3). Se provassimo a calcolare la mutua

informazione tra una variabile e sé stessa ci aspetteremmo di trovare un valore in qualche modo massimale, giacché è ragionevole che il massimo grado di dipendenza si sviluppi proprio in questo caso. Per valutare $I(X \wedge X)$ bisogna conoscere la matrice della d.p. congiunta, che si ricava subito non appena si consideri che $p(x_i, x_j) = 0$ se $i \neq j$, mentre $p(x_i, x_i) = p(x_i)$. Sostituendo nella (2.5) e supponendo che sia $1 \leq i \leq K$) ricaviamo

$$I(X \wedge X) = \sum_{i,j} p(x_i, x_j) \log \frac{p(x_i, x_j)}{p(x_i)p(x_j)} = -\sum_{i=1}^{K} p(x_i) \log p(x_i) \geq 0 \quad (2.6)$$

L'espressione che rimane definita sulla destra della (2.6) è la celeberrima *entropia di Shannon* $H(X)$, che esprime anche la speranza matematica della v.a. $\mathcal{I}(X) = -\log \Pr\{X\}$, detta anche *autoinformazione*; dunque

$$H(X) \triangleq E[-\log \Pr\{X\}] = -\sum_{i=1}^{K} p(x_i) \log p(x_i) \quad (2.7)$$

Da un punto di vista storico il processo si è svolto in modo opposto, poiché già nel 1928 Hartley propose una misura *logaritmica* per l'auto-informazione $\mathcal{I}(X)$ associata alla realizzazione di un evento x che possiede una probabilità a priori $p(x)$ di verificarsi. Una misura logaritmica esprime bene la circostanza che è ragionevole considerare nulla la quantità d'informazione associata alla realizzazione di eventi *certi* ($p(x) = 1$); è inoltre altrettanto ragionevole ipotizzare un'informazione che tenda a essere infinitamente grande quando la probabilità a priori tende a 0 (si veda il diagramma di figura 2.1). Naturalmente sono mol-

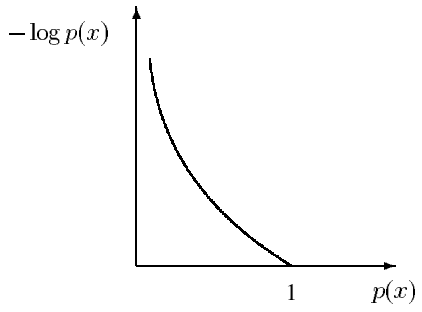


Figura 2.1 Diagramma della funzione di autoinformazione.

te le funzioni che posseggono un comportamento qualitativo analogo a quello dell'autoinformazione logaritmica; tuttavia, come già anticipato, vedremo che il

logaritmo è l'unica funzione a soddisfare una proprietà (della *diramazione*) che risulta cruciale in una definizione assiomatica di una misura d'informazione (si veda la sezione 2.4). Poiché l'entropia è il valor medio dell'auto-informazione, possiamo pensare che essa esprima la *quantità media d'informazione* associata alla realizzazione di una v.a. o anche la quantità (media) d'incertezza rimossa dalla realizzazione della stessa variabile aleatoria. In una prospettiva diversa la si può interpretare anche come informazione necessaria a specificare la realizzazione (di una v.a.) o quantità d'incertezza *a priori* sull'esperimento.

Riprendiamo ora in considerazione la (2.5) e scomponiamo la mutua informazione mediante il teorema di Bayes, sulla base della relazione $p(x,y) = p(y)p(x/y)$. Si ottiene

$$\begin{aligned} I(X \wedge Y) &= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = \sum_{x,y} p(x,y) \log \frac{p(x/y)}{p(x)} = \\ &= -\sum_{x} \left(\sum_{y} p(x,y) \right) \log p(x) + \sum_{x,y} p(x,y) \log p(x/y) = \\ &= H(X) - \left(-\sum_{x,y} p(x,y) \log p(x/y) \right) \end{aligned} \tag{2.8}$$

dove l'ultima uguaglianza segue dal fatto che $\sum_y p(x,y) = p(x)$. Il termine di destra sull'ultima riga rappresenta la speranza matematica $E[-\log \Pr\{X/Y\}]$ della variabile X *condizionata alla conoscenza* della variabile Y. Si tratta dunque di un *entropia condizionata* $H(X/Y)$. Detta entropia può essere espressa anche nei termini della $p(y)$:

$$\begin{aligned} H(X/Y) \triangleq -\sum_{x,y} p(x,y) \log p(x/y) &= \sum_{y} p(y) \left(-\sum_{x} p(x/y) \log p(x/y) \right) = \\ &= \sum_{y} p(y) H(X/Y = y) \end{aligned} \tag{2.9}$$

Con analoga definizione il condizionamento si può estendere a più variabili; in tal caso l'entropia condizionata diventa

$$H(X/Y,Z) = -\sum_{x,y,z} p(x,y,z) \log p(x/y,z) \tag{2.10}$$

Alla luce di tutto ciò e tenendo conto della simmetria tra X e Y possiamo riscrivere la (2.8) come

$$I(X \wedge Y) = H(X) - H(X/Y) = H(Y) - H(Y/X) \geq 0 \tag{2.11}$$

il che consente di esprimere la mutua informazione anche come quantità media d'incertezza rimossa su X (Y) dalla conoscenza di Y (X). Dall'ultima disuguaglianza della (2.11) e dal fatto che l'entropia è una quantità sempre non negativa, si deducono le seguenti relazioni:

$$\begin{aligned} &H(X) \geq H(X/Y) \quad H(Y) \geq H(Y/X) \quad (= \text{ sse le v.a. sono indip.}) \\ &I(X \wedge Y) \leq H(X) \quad I(X \wedge Y) \leq H(Y) \quad (= \text{ sse } \quad Y = f(X)) \\ &\qquad I(X \wedge Y) \leq \min\{H(X), H(Y)\} \end{aligned} \tag{2.12}$$

stabilendo così che la conoscenza di una certa v.a. condizionante Y non può comportare un incremento dell'incertezza media relativa a un'altra v.a. X (comporta semmai una diminuzione). Ciò significa che aggiungere variabili condizionanti porta eventualmente a una diminuzione dell'entropia

$$H(X) \geq H(X/Y) \geq H(X/Y,Z) \geq H(X/Y,Z,K) \geq \ldots \tag{2.13}$$

Nella catena di disuguaglianze sopra riportate l'uguaglianza sussiste se e solo se c'è indipendenza stocastica tra le variabili; in quest'ultimo caso la conoscenza della variabili condizionanti non riduce l'incertezza relativa alla X, come è naturale che sia. Inoltre, dal confronto tra la (2.12) e la (2.6) si evince che la mutua informazione raggiunge il suo valore massimo (l'entropia) quando si confronta una v.a. con sé stessa. Se la Y risulta essere una *funzione deterministica* della X si ha infine $H(Y/X) = 0$.

Il condizionamento può riguardare anche la mutua informazione

$$\begin{aligned} I(X \wedge Y/Z) &= \sum_{x,y,z} p(x,y,z) \log \frac{p(x,y/z)}{p(x/z)p(y/z)} = \\ &= H(X/Z) - H(X/ZY) = H(Y/Z) - H(Y/ZX) \geq 0 \end{aligned} \tag{2.14}$$

con ovvio significato dei simboli.

Oss. 2.1. L'indipendenza tra v.a. non condizionate *non* comporta l'indipendenza tra le stesse v.a. condizionate (alla conoscenza di una terza v.a.), e viceversa. In altre parole può accadere che sia $I(X \wedge Y) = 0$, ma $I(X \wedge Y/Z) > 0$ strettamente, oppure $I(X \wedge Y) > 0$ con $I(X \wedge Y/Z) = 0$. Per il primo caso si pensi a una situazione in cui X, Y e Z siano v.a. binarie tali che $Z = X \oplus Y$, con $\oplus$ somma modulo 2, mentre per il secondo caso rimandiamo alla sezione 2.5, nella quale si descriveranno le proprietà relative a v.a. poste in *catena di Markov* (cioè $X \to Y \to Z$ con Z che dipende da X *solo* attraverso Y).

2.3 Altre proprietà dell'entropia e della mutua informazione

Per quanto riguarda le altre proprietà formali dell'entropia si può verificare facilmente che

$$0 \leq H(X) \leq \log |\mathcal{X}| \qquad \begin{matrix} (=0 \quad \text{sse } P_X \text{ è degenere}) \\ (=\log |\mathcal{X}| \quad \text{sse } P_X \text{ è uniforme}) \end{matrix} \tag{2.15}$$

dove $|\mathcal{X}|$ è la cardinalità dell'insieme $\mathcal{X}$. La disuguaglianza a sinistra deriva direttamente dalla (2.6), o più semplicemente dal fatto che l'entropia è somma di termini tutti positivi; l'uguaglianza a zero la si ha inoltre nel solo caso in cui la d.p. associata sia *degenere*, cioè nella forma $0, 0, \ldots, 1, \ldots, 0$, in cui tutte le probabilità sono nulle, tranne una che vale 1. Il valore massimo dell'entropia, e dunque l'incertezza massima a priori, la si ottiene quando si è di fronte a una d.p. *uniforme*, in cui se $|\mathcal{X}| = K$ si ha $p(x_i) = 1/K \ \forall i$. La verifica della limitazione superiore per la (2.15) è immediata, non appena si calcoli la divergenza informazionale tra la P_X e la d.p. uniforme $U = \{1/K\}_{i=1}^{K}$:

$$D(P_X /\!/ U) = \sum_{i=1}^{K} p(x_i) \log [p(x_i) K] = \log K - H(X) \geq 0$$

Poiché è possibile associare un'entropia a qualunque d.p., possiamo definire anche una *entropia congiunta* delle variabili X e Y

$$H(X,Y) = -\sum_{x,y} p(x,y) \log p(x,y) \tag{2.16}$$

che s'interpreta in modo analogo come quantità media d'informazione associata alla realizzazione della v.a. congiunta (X,Y). Usando al solito il teorema di Bayes per sciogliere la distribuzione congiunta si ottiene

$$\begin{aligned} H(X,Y) &= -\sum_{x,y} p(x,y) \log p(x,y) = \\ &= -\sum_{x,y} p(x,y) \log p(x) - \sum_{x,y} p(x,y) \log p(y/x) = \\ &= H(X) + H(Y/X) = H(Y) + H(X/Y) \end{aligned} \tag{2.17}$$

dove l'ultima disuguaglianza segue dalla simmetria tra X e Y. Si noti che la (2.17) costituisce una reintepretazione del teorema di Bayes in termini di entropia: l'incertezza associata alla congiunta (X,Y) è pari all'incertezza su X più quella su Y noto X. Fra l'altro, essendo le entropie condizionate quantità non negative, resta stabilito anche che

$$H(X,Y) \geq \begin{cases} H(X) \\ H(Y) \end{cases} \quad \text{cioè} \quad H(X,Y) \geq \max\{H(X), H(Y)\} \tag{2.18}$$

come dev'essere, poiché è ragionevole che l'incertezza su due v.a. sia non minore di quella relativa a una sola delle due. Anche in questo caso $H(X,Y) = H(X)$ se e solo se la Y è funzione deterministica della X. Coniugando la (2.18) con la (2.13) si ottiene

$$\begin{aligned} \ldots \geq H(X,Y,Z) \geq H(X,Y) \geq H(X) \geq \\ \geq H(X/Y) \geq H(X/Y,Z) \geq H(X/Y,Z,K) \geq \ldots \end{aligned} \tag{2.19}$$

Un'ultima osservazione riguarda la presenza nell'entropia di v.a. che siano funzioni deterministiche di qualche altra v.a. Per stabilire il legame che si crea tra la variabile X e una sua funzione deterministica $f(X)$ si può scomporre $H(X, f(X))$ secondo la (2.17)

$$H(X, f(X)) = \begin{cases} H(X) + H(f(X)/X) \\ H(f(X)) + H(X/f(X)) \end{cases} \tag{2.20}$$

Si può verificare facilmente che il termine $H(f(X)/X)$ è nullo, come appare del resto anche euristicamente (noto X non si ha alcuna incertezza residua su $f(X)$, che è sua funzione *deterministica*). Dal confronto tra le due esplicitazioni della $H(X, f(X))$ si deduce

$$H(X) = H(f(X)) + H(X/f(X))$$

dalla quale segue una una relazione generale che lega l'incertezza di una v.a. con quella relativa a una sua funzione deterministica

$$H(f(X)) \leq H(X) \qquad (= \text{ sse biunivoca}) \tag{2.21}$$

valendo l'uguaglianza nel caso di biunivocità della funzione. Se la variabile che è funzione deterministica risulta condizionante, posto $g(Y) = (Y, f(Y))$ funzione biunivoca possiamo scrivere

$$H(X/f(Y)) \geq H(X/Y, f(Y)) = H(X/g(Y)) = H(X/Y) \tag{2.22}$$

valendo l'uguaglianza in caso di univocità della f. Anche in questo caso si ha una conferma intuitiva della disuguaglianza, poiché appare ragionevole ipotizzare che l'incertezza su X sia più grande quando si conosca *solo* $f(Y)$ piuttosto che Y; infatti $f(Y)$ porta con sé meno informazione *condizionante* (per convincersi di ciò si pensi al caso limite in cui sia $f(Y) = \text{cost}$). Vediamo ora un esempio riassuntivo.

Esempio 2.1. [19] Sia assegnata la seguente tabella per la distribuzione P_{XY} delle probabilità congiunte $p(x, y)$.

	X	1/2	1/4	1/8	1/8
Y	P_{XY}	x_1	x_2	x_3	x_4
1/4	y_1	1/8	1/16	1/32	1/32
1/4	y_2	1/16	1/8	1/32	1/32
1/4	y_3	1/16	1/16	1/16	1/16
1/4	y_4	1/4	0	0	0

Le marginali sono rispettivamente $P_X = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$ e $P_Y = \{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\}$.
Calcoliamo le varie entropie associate allo schema

$$H(X) = -\sum_{i=1}^{4} p(x_i) \log p(x_i) = \frac{1}{2} \log 2 + \frac{1}{4} \log 4 + 2 \cdot \frac{1}{8} \log 8 = \frac{7}{4}$$

$$H(Y) = -\sum_{i=1}^{4} p(y_i) \log p(y_i) = 4 \cdot \frac{1}{4} \log 4 = 2$$

$$\begin{aligned} H(X/Y) &= -\sum_{i=1}^{4} p(Y = y_i) H(X/Y = y_i) = \\ &= \frac{1}{4} H\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right) + \frac{1}{4} H\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{8}, \frac{1}{8}\right) + \\ &+ \frac{1}{4} H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) + \frac{1}{4} H\left(1, 0, 0, 0\right) = \frac{11}{8} \end{aligned}$$

$$\begin{aligned} H(Y/X) &= -\sum_{i=1}^{4} p(X = x_i) H(Y/X = x_i) = \\ &= \frac{1}{2} H\left(\frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{2}\right) + \frac{1}{4} H\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}, 0\right) + \\ &+ \frac{1}{8} H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, 0\right) + \frac{1}{8} H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, 0\right) = \frac{13}{8} \end{aligned}$$

$$H(X, Y) = H(X) + H(Y/X) = H(Y) + H(X/Y) = \frac{27}{8}$$

$$I(X \wedge Y) = H(X) - H(X/Y) = H(Y) - H(Y/X) = \frac{3}{8}$$

○

È possibile specificare meglio il legame che esiste nella (2.19) tra variabili condizionanti e non introducendo una generalizzazione della (2.17). È questo il caso dell'entropia di un *vettore* $\boldsymbol{X^n} = (X_1, X_2, \ldots, X_n)$ di v.a., le cui realizzazioni $\boldsymbol{x}$ prendono valori sull'insieme prodotto $\boldsymbol{\mathcal{X}^n}$. Se $H(\boldsymbol{X^n}) = H(X_1, X_2, \ldots, X_n) = -\sum_{\boldsymbol{x}\in\mathcal{X}^n} p(\boldsymbol{x}) \log p(\boldsymbol{x})$, il legame con le sue componenti viene specificato dal seguente

Teorema 2.3 (Regola della catena). *Per $i \geq 2$ vale la seguente relazione*

$$H(\boldsymbol{X^n}) = H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i/X_1, X_2, \ldots, X_{i-1}) \qquad (2.23)$$

Dim. La dimostrazione può derivare dall'applicazione iterativa della (2.17) ai sottovettori $X_1, X_2, \ldots, X_i$ oppure, meglio ancora, dall'impiego dell'induzione. Per $n = 1$ la formula è banalmente verificata. Supponiamo lo sia per $n-1$ e dimostriamone la validità per n

$$\begin{aligned} H(X_1, X_2, \ldots, X_n) &= H(X_1, X_2, \ldots, X_{n-1}) + H(X_n/X_1, X_2, \ldots, X_{n-1}) = \\ &= \sum_{i=1}^{n-1} H(X_i/X_1, X_2, \ldots, X_{i-1}) + H(X_n/X_1, X_2, \ldots, X_{n-1}) = \\ &= \sum_{i=1}^{n} H(X_i/X_1, X_2, \ldots, X_{i-1}) \end{aligned}$$

dove la seconda uguaglianza deriva dall'applicazione dell'ipotesi induttiva. □

Si noti che per effetto della (2.13) possiamo scrivere

$$H(\boldsymbol{X^n}) = H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i) \qquad (2.24)$$

valendo l'uguaglianza se e solo se le componenti sono *indipendenti*.

Un'altra relazione notevole riguarda il rapporto tra variabili vettoriali e le sue componenti in presenza di condizionamento. Se $\boldsymbol{X^n} = (X_1, X_2, \ldots, X_n)$ è una v.a. vettoriale condizionante, possiamo sfruttare la regola della catena per scrivere

$$\begin{aligned} H(\boldsymbol{Y^n}/\boldsymbol{X^n}) &= H(Y_1, Y_2, \ldots, Y_n/\boldsymbol{X^n}) = \\ &= \sum_{i=1}^{n} H(Y_i/\boldsymbol{X^n}, Y_1, Y_2, \ldots, Y_{i-1}) \leq \sum_{i=1}^{n} H(Y_i/X_i) \end{aligned} \qquad (2.25)$$

Si può verificare facilmente (p.es. scrivendo l'espressione esplicita dell'entropia condizionata) che l'uguaglianza vale se e solo se

$$p(\boldsymbol{y}/\boldsymbol{x}) = p(y_{i_1}, y_{i_2}, \ldots, y_{i_n}/x_{j_1}, x_{j_2}, \ldots, x_{j_n}) = \prod_{r=1}^{n} p(y_{i_r}/x_{j_r}) \qquad (2.26)$$

Questa condizione ha un preciso significato nell'ambito del modello di canale *stazionario e senza memoria* (sezione 4.2), nel quale $\boldsymbol{X^n}$ e $\boldsymbol{Y^n}$ sono rispettivamente le v.a. (vettoriali) d'ingresso e d'uscita del canale.

Studiamo ora il rapporto tra la mutua informazione di v.a. vettoriali $I(\boldsymbol{X^n} \wedge \boldsymbol{Y^n})$ e la mutua informazione delle componenti $I(X_i \wedge Y_i)$. Ci sono essenzialmente due casi particolari che vale la pena analizzare

Teorema 2.4. *Se le componenti X_i sono indipendenti vale la relazione*

$$I(\boldsymbol{X^n} \wedge \boldsymbol{Y^n}) \geq \sum_{i=1}^{n} I(X_i \wedge Y_i) \tag{2.27}$$

Dim.

$$I(\boldsymbol{X^n} \wedge \boldsymbol{Y^n}) = H(\boldsymbol{X^n}) - H(\boldsymbol{X^n}/\boldsymbol{Y^n}) = \sum_{i=1}^{n} H(X_i) - H(\boldsymbol{X^n}/\boldsymbol{Y^n}) \geq$$
$$\geq \sum_{i=1}^{n} H(X_i) - \sum_{i=1}^{n} H(X_i/Y_i) = \sum_{i=1}^{n} I(X_i \wedge Y_i)$$

dove la seconda uguaglianza deriva dall'ipotesi di indipendenza per le X_i e la disuguaglianza dalla (2.25) (con X e Y scambiate). Inoltre l'uguaglianza la si ha nel solo caso in cui le $Y_1, Y_2, \ldots, Y_n$ siano tra loro indipendenti □

L'altro caso particolare è relativo a un'ipotesi che specificheremo meglio nel prossimo capitolo (si veda la (2.26))

Teorema 2.5. *Se $p(\boldsymbol{y}/\boldsymbol{x}) = p(y_{i_1}, y_{i_2}, \ldots, y_{i_n}/x_{j_1}, x_{j_2}, \ldots, x_{j_n}) = \prod_{r=1}^{n} p(y_{i_r}/x_{j_r})$ allora*

$$I(\boldsymbol{X^n} \wedge \boldsymbol{Y^n}) \leq \sum_{i=1}^{n} I(X_i \wedge Y_i) \tag{2.28}$$

Dim.

$$I(\boldsymbol{X^n} \wedge \boldsymbol{Y^n}) = H(\boldsymbol{Y^n}) - H(\boldsymbol{Y^n}/\boldsymbol{X^n}) = H(\boldsymbol{Y^n}) - \sum_{i=1}^{n} H(Y_i/X_i) \leq$$
$$\leq \sum_{i=1}^{n} H(Y_i) - \sum_{i=1}^{n} H(Y_i/X_i) = \sum_{i=1}^{n} I(X_i \wedge Y_i)$$

dove la seconda uguaglianza deriva dalla (2.25) quando vale l'ipotesi del teorema, mentre la disuguaglianza deriva dalla (2.24). L'uguaglianza sussiste quando le Y_i sono indipendenti □

Quando valgono contemporaneamente le ipotesi dei teoremi 2.4 e 2.5 si ricava

$$I(\boldsymbol{X^n} \wedge \boldsymbol{Y^n}) = \sum_{i=1}^{n} I(X_i \wedge Y_i) \tag{2.29}$$

L'ultimo risultato che tratteremo in questa sezione è la celebre *disuguaglianza di Fano*. Il contesto è quello in cui due v.a., X e $\widehat{X}$, assumono i loro valori su un insieme finito $\mathcal{X}$, con $|\mathcal{X}| = K$ elementi. Spesso la $\widehat{X}$ sarà interpretata come *stima* della X. Quando ciò accade poniamo $\epsilon = \Pr\{X \neq \widehat{X}\}$, e dunque ϵ rappresenta la probabilità che la stima $\widehat{X}$ di X sia errata. Una situazione di questo tipo è molto frequente non solo in statistica, ma anche nella codifica su un *canale rumoroso* (vedi 4.2), dove la $\widehat{X}$ può derivare da un processo di *decodifica* nel quale si cerca appunto di stimare la lettera che è stata trasmessa. Vedremo un'applicazione della disuguaglianza di Fano anche nell'ambito della crittografia (7.4.1). La disuguaglianza rappresenta una limitazione superiore per l'incertezza che si ha su X quando sia nota una sua stima.

Teorema 2.6 (Disuguaglianza di Fano). *Posto* $h(\epsilon) = -\epsilon \log \epsilon - (1-\epsilon)\log(1-\epsilon)$ *si ha*

$$H(X/\widehat{X}) \leq h(\epsilon) + \epsilon \log(K-1) \tag{2.30}$$

Dim. Per la dimostrazione introduciamo una v.a. Z di supporto, di tipo binario, che segnala la presenza di un errore:

$$Z = \begin{cases} 1 & \text{se } X \neq \widehat{X} \qquad \text{con} \qquad \epsilon = \Pr\{X \neq \widehat{X}\} = \Pr\{Z=1\} \\ 0 & \text{se } X = \widehat{X} \qquad \text{con} \quad 1-\epsilon = \Pr\{X = \widehat{X}\} = \Pr\{Z=0\} \end{cases}$$

Procediamo ora sfruttando la legge di composizione delle entropie

$$\begin{aligned} H(X, Z/\widehat{X}) &= H(X/\widehat{X}) + H(Z/X, \widehat{X}) = \\ &= H(Z/\widehat{X}) + H(X/Z, \widehat{X}) \end{aligned}$$

Dal confronto tra i due termini, e dal fatto che $H(Z/X, \widehat{X}) = 0$ (noti X e $\widehat{X}$ non c'è incertezza residua su Z), si deduce

$$\begin{aligned} H(X/\widehat{X}) &= H(Z/\widehat{X}) + H(X/Z, \widehat{X}) \leq \\ &\leq H(Z) + \Pr\{Z=1\} H(X/Z=1, \widehat{X}) + \\ &\qquad + \Pr\{Z=0\} H(X/Z=0, \widehat{X}) \leq \\ &\leq h(\epsilon) + \epsilon \log(K-1) \end{aligned}$$

dove la prima disuguaglianza deriva dalla (2.19). Per la seconda si noti innanzitutto che $H(X/Z=0, \widehat{X}) = 0$ (se $Z=0$ si ha $X = \widehat{X}$), e che $H(Z)$ è in effetti

la $h(\epsilon)$. Inoltre, se $Z = 1$ sappiamo che $X \neq \widehat{X}$, e dunque nota la stima, X può assumere solo uno tra i restanti $K-1$ valori; l'entropia viene allora limitata superiormente da $\log(K-1)$. $\square$

Della disuguaglianza di Fano si può fornire anche un'interpretazione di tipo euristico: l'incertezza media su X nota una sua stima la si può determinare valutando innanzitutto se X e $\widehat{X}$ coincidono; il valore medio di tale incertezza è $h(\epsilon)$. Se non coincidono, e ciò succede con probabilità ϵ, rimane un'incertezza residua sulle $K-1$ realizzazioni rimanenti, limitata superiormente da $\log(K-1)$. La figura 2.2 illustra l'andamento della funzione $h(\epsilon) + \epsilon \log(K-1)$; in particolare quando $H(X/\widehat{X}) \geq \eta > 0$ strettamente, la validità del teorema 2.6 implica che anche la probabilità ϵ debba essere discosta dallo zero. In altre parole abbiamo $0 < \epsilon^* \leq \epsilon$, dove l'ascissa ϵ^* resta determinata dall'intersezione tra la funzione e la retta orizzontale con ordinata η.

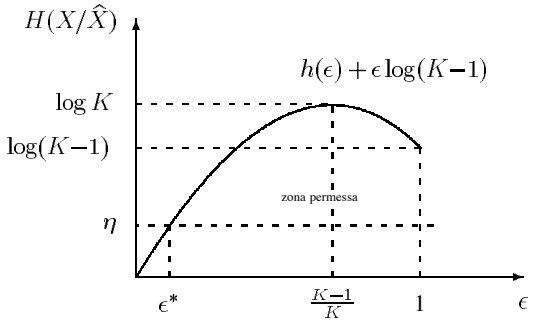


Figura 2.2 Diagramma della funzione $h(\epsilon) + \epsilon \log(K-1)$.

2.4 Caratterizzazione assiomatica dell'entropia

Concentriamo ora la nostra attenzione sul problema di fornire una caratterizzazione dell'entropia basata sulle delle proprietà naturali che una misura d'informazione dovrebbe possedere.

Il termine *entropia* venne coniato nel 1864 da Clausius, nel contesto della termodinamica. Esso venne ripreso successivamente da Shannon nel 1948, nella definizione di una misura d'informazione intesa come valor medio dell'autoinformazione. L'entropia di Shannon, così come definita, fece subito risaltare la sua centralità nelle operazioni di codifica per sorgenti stazionarie e senza memoria, risultando in qualche modo legata, nel teorema di codifica per dette

sorgenti, al *tasso* del codice (si veda il par.3.4.6). Tutto ciò è ampiamente sufficiente per giudicarla, a buon diritto, come uno degli interpreti principali della teoria dell'informazione. La posizione di rispettabilità di detta funzione venne se possibile esaltata nel corso di alcuni studi successivi, orientati essenzialmente a definire una misura d'informazione per via assiomatica. Rimandiamo il lettore interessato ai dettagli al testo di Aczél-Daróczy [1]. In questa sede ci limitiamo a ricordare che è possibile *dimostrare* l'unicità dell'entropia di Shannon tra tutte le funzioni per le quali si richieda la validità di un certo insieme di *postulati*, i quali delineano le proprietà naturali che una misura d'informazione dovrebbe ragionevolmente possedere. Detta $\mathcal{H}_K$ una generica funzione nelle variabili $p_1, p_2, \ldots, p_K$, candidata a diventare una misura d'informazione, è lecito attendersi che $\mathcal{H}$ sia *continua*, poiché in corrispondenza di piccole variazioni delle p_i è ragionevole che anche $\mathcal{H}_K$ vari di poco. Si richiede inoltre che la funzione sia *simmetrica* nelle sue variabili, cioè che il suo valore non cambi permutando i pedici delle variabili. Ma la vera proprietà discriminante è quella della *diramazione*, già ricordata in precedenza, e che riguarda la perdita d'informazione che si soffre quando non si discriminano più due eventi distinti. L'insieme di postulati che si possono usare non è univocamente determinato; tuttavia quelli che riportiamo nel seguito sono tra i più intuitivi e più usati (anche se non costituiscono un insieme *minimo*). Vediamoli nei dettagli.

P1) Continuità: $\mathcal{H}_2(p, 1-p)$ è continua in p;

P2) Normalizzazione: $\mathcal{H}_2(1/2, 1/2) = 1$;

P3) Simmetria: $\mathcal{H}_K(p_1, p_2, \ldots, p_K)$ è simmetrica nelle p_i;

P4) Diramazione: $\mathcal{H}_K(p_1, p_2, \ldots, p_K) =$

$$= \mathcal{H}_{K-1}(p_1 + p_2, p_3, \ldots, p_K) + (p_1 + p_2)\mathcal{H}_2\left(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2}\right)$$

L'entropia di Shannon (2.7) soddisfa banalmente i postulati P1) - P3). Per la verifica di P4) è sufficiente aggiungere e togliere a $H(p_1, p_2, \ldots, p_K)$ il termine $(p_1 + p_2)\log(p_1 + p_2)$.

La validità del postulato P4) sottende in realtà una generalizzazione che consente di esprimere la perdita d'informazione anche quando si aggreghino più sottoinsiemi di eventi. Si supponga infatti di partire da un'entropia K-aria $H_K(p_1, p_2, \ldots, p_K)$ e di non distinguere più gli eventi i cui indici appartengono a determinati sottoinsiemi $\mathcal{J}_i, 1 \leq i \leq M$ con $M < K$. Di conseguenza viene definita una nuova d.p. $q_1, q_2, \ldots, q_M$, dove le probabilità q_i derivano dalla somma delle probabilità relative agli eventi che sono stati aggregati, e che appaiono ora indistinguibili, cioè $q_i = \sum_{j \in \mathcal{J}_i} p_j$. Cerchiamo ora di esprimere

$H_K(p_1, p_2, \ldots, p_K)$ in funzione di $H_M(q_1, q_2, \ldots, q_M)$:

$$H_K(p_1, p_2, \ldots, p_K) = -\sum_{r=1}^{K} p_r \log p_r = -\sum_{i=1}^{M} \sum_{j \in \mathcal{J}_i} \frac{p_j}{q_i} q_i \log \frac{p_j}{q_i} q_i =$$

$$= -\sum_{i=1}^{M} q_i \sum_{j \in \mathcal{J}_i} \left(\frac{p_j}{q_i}\right) \log \left(\frac{p_j}{q_i}\right) - \sum_{i=1}^{M} q_i \log q_i \left(\sum_{j \in \mathcal{J}_i} \frac{p_j}{q_i}\right) =$$

$$= \sum_{i=1}^{M} q_i H^{(i)} + H_M(q_1, q_2, \ldots, q_M)$$

dove $H^{(i)} \triangleq -\sum_{j \in \mathcal{J}_i} \left(\frac{p_j}{q_i}\right) \log \left(\frac{p_j}{q_i}\right)$ è l'entropia associata a $\mathcal{J}_i$ $\left(\sum_{j \in \mathcal{J}_i} \frac{p_j}{q_i} = 1\right)$ e dunque la p_j/q_i è effettivamente una d.p.. Possiamo dunque scrivere

$$H_K(p_1, p_2, \ldots, p_K) - H_M(q_1, q_2, \ldots, q_M) = \sum_{i=1}^{M} q_i H^{(i)} \tag{2.31}$$

La relazione esprime la perdita d'informazione media che si soffre quando alcuni sottoinsiemi di eventi vengono raggruppati e resi indistinguibili; tale perdita è data dalla somma delle entropie relative ai sottoinsiemi pesati con le probabilità degli stessi. Questa espressione generale della proprietà della diramazione ha un'importanza rilavante nell'ambito delle applicazioni, anche al di fuori del contesto della teoria dell'informazione. Infatti in tutti i casi in cui esista una K-pla di interi (≥ 0) $N_1, N_2, \ldots, N_K$ è possibile effettuare una normalizzazione ponendo $p_i = N_i/N$, dove $N = \sum_{i=1}^{K} N_i$. Ciò consente di agganciarsi a una d.p., interpretando l'entropia direttamente come *indice di frammentazione* della K-pla; infatti se tutti gli N_i sono uguali la frammentazione (l'entropia) è massima, poiché la d.p. indotta è uniforme. Viceversa quando c'è un forte squilibrio tra i valori (al limite uno solo è maggiore di zero e tutti gli altri sono nulli) si tende verso la massima omogeneità, caratterizzata da una d.p. (quasi) degenere e da un'entropia (quasi) nulla.

La descrizione dell'entropia su base assiomatica può essere arricchita introducendo un modello in cui gli eventi vengano considerati non solo sulla base delle loro proprietà statistiche, ma anche tenendo conto di un parametro u_i legato all'importanza dell'evento i-esimo, cioè alla sua *utilità* con riferimento a un qualche scopo. Individuata dunque una K-pla di reali $\mathcal{U} = \{u_1, u_2, \ldots, u_K\}$, $u_i \geq 0$, che caratterizzano le utilità degli eventi di probabilità p_i, possiamo definire in modo naturale un'*entropia utile* (di Beliş-Guiaşu) mediante l'espressione

$$H(P;\mathcal{U}) = -\sum_{i=1}^{K} u_i p_i \log p_i \tag{2.32}$$

Si può verificare facilmente che l'assioma P4) continua a valere, a patto che si definisca in modo opportuno l'utilità della composizione di due eventi e_1, e_2, cioè come media pesata delle utilità dei singoli eventi:

$$\text{Util}(e_1, e_2) = u_{12} = \frac{p_1}{p_1 + p_2} u_1 + \frac{p_2}{p_1 + p_2} u_2$$

La P4) si può allora esprimere nel modo seguente ($p_{12} = p_1 + p_2$):

$$H_K(p_1, p_2, \ldots, p_K; u_1, u_2, \ldots, u_K) =$$
$$H_{K-1}(p_{12}, p_3 \ldots, p_K; u_{12}, u_3, \ldots, u_K) + p_{12} H_2(\frac{p_1}{p_{12}}, \frac{p_2}{p_{12}}; u_1, u_2) \quad (2.33)$$

La presenza dei coefficienti u_i comporta l'annullamento dell'entropia utile quando essi sono tutti nulli. La (2.32) si annulla anche quando gli eventi utili sono impossibili o gli eventi possibili sono inutili. L'ultimo caso di annullamento lo si ha quando P è degenere.

Quest'ultima evenienza gode di un'interpretazione particolarmente significativa in un ambito applicativo nel quale più *strumenti di misura*, caratterizzati da una accuratezza u_i e da un'affidabilità (normalizzata) p_i, concorrono in modo sinergico alla valutazione di una qualche grandezza derivata G. Il fatto che l'informazione utile si annulli a seguito di d.p. degeneri significa che non si considera globalmente affidabile una valutazione della grandezza G che derivi dall'impiego di *un unico* strumento, anche se questi fornisce una misurazione potenzialmente molto accurata. In altre parole viene maggiormente accreditata una misura che privilegi un'integrazione sinergica dei dati più che una loro valutazione, anche molto accurata, derivante però da un singolo strumento di misura.

2.5 Teoremi di elaborazione dei dati

I teoremi che tratteremo in questa sezione sono d'importanza fondamentale per tutti quei sistemi in cui l'informazione venga elaborata da blocchi funzionali concatenati (o in *cascata*, seguendo il linguaggio tecnico-ingegneristico). In un certo senso essi costituiscono una sorta di analogia (in ambito informazionale) col secondo principio della termodinamica. L'idea di base è intuitiva: l'elaborazione dei dati è un processo essenzialmente *irreversibile*, che porta a un degrado dell'informazione. Questo degrado ha un'interpretazione entropica, nel senso che ogni elaborazione è accompagnata da un (eventuale) aumento dell'entropia, e dunque dell'incertezza associata alle variabili che sono state elaborate.

Il modello che prenderemo in esame prevede un insieme di v.a. concatenate, diciamo $X, Y, Z, \ldots$, che assumono i loro valori sugli insiemi finiti $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \ldots$, nel quale ciascuna variabile dipende dalla variabile precedente *solo* attraverso

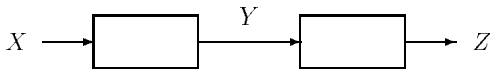


Figura 2.3 Variabili X, Y, Z in catena di Markov.

una terza variabile che sta fra le due nella catena. Con riferimento alle v.a. di cui sopra diciamo che Z dipende da X *solo attraverso* Y, e scriviamo $X \to Y \to Z$. Quando ciò accade si dice che le variabili sono in *catena di Markov*. Se si preferisce si può pensare a due blocchi funzionali disposti in cascata (vedi figura 2.3), in cui la v.a. Y uscita del primo blocco alimenta l'ingresso del secondo. In tal caso l'uscita Z del secondo blocco dipende solo dall'ingresso del blocco (Y). Questo implica una concatenazione stretta tra le variabili e in particolare la non esistenza di "corto circuiti" informazionali tra X e Z. Un altro modo per esprimere intuitivamente la condizione è quello di dire che tutta l'informazione generata da X *passa* attraverso Y (sia pure come risultato di una elaborazione). Dal punto di vista matematico la condizione viene fissata dalla validità della seguente relazione probabilistica (e dalla sua controparte informazionale)

$$p(z/x,y) = p(z/y) \qquad\qquad H(Z/X,Y) = H(Z/Y) \tag{2.34}$$

che porta immediatamente a $H(X,Y,Z) = H(X) + H(Y/X) + H(Z/Y)$. Si noti peraltro che, essendo $I(X \wedge Z/Y) = H(Z/Y) - H(Z/X,Y) = 0$, la conoscenza di Y annulla la mutua informazione condizionata, anche se in genere X e Z non sono indipendenti (al contrario ci si aspetta una forte correlazione tra due variabili legate da un processo di elaborazione; si veda a tal proposito l'osservazione 2.1). Ciò si spiega, nel modello delle variabili in catena, affermando che la conoscenza di Y *assorbe* tutto il flusso d'informazione tra X e Z, cosicché le due variabili non hanno più informazione da scambiare. La catena di Markov non è però unidirezionale, come specifica il seguente

Lemma 2.1. *Se $X \to Y \to Z$ allora vale anche la catena opposta $Z \to Y \to X$.*

Dim. Immediata non appena si esprima l'uguaglianza a zero della mutua informazione condizionata $I(X \wedge Z/Y)$ come $H(X/Y) - H(X/Z,Y) = 0$. □

L'ipotesi di catena di Markov per le variabili X, Y, Z porta ai seguenti teoremi.

Teorema 2.7 (I teorema di elaborazione dei dati). *Se* $X \to Y \to Z$ *allora*

$$H(X/Y) \leq H(X/Z) \tag{2.35}$$

Dim. $H(X/Y) = H(X/Z,Y) \leq H(X/Z)$ dove la prima uguaglianza segue dalla dimostrazione del lemma (2.1) e la disuguaglianza dalle ordinarie proprietà dell'entropia condizionata. □

La giustificazione euristica del teorema è immediata: se si vuole conoscere la X conviene "interrogare" Y piuttosto che Z, che deriva da Y sulla base di una elaborazione potenzialmente irreversibile. In altre parole la Z, che è una versione "degradata" della Y, informa su X meno di quanto informi Y. Si osservi che ciò esprime un importante e ben preciso limite normativo alle potenzialità offerte da una elaborazione dell'informazione, per quanto intelligente e sofisticata possa essere. In altri termini l'informazione persa su X a seguito dell'irreversibilità di una sua elaborazione non è più recuperabile, per quanto "astuta" possa essere l'elaborazione successiva. Il caso di uguaglianza vale quando $H(X/Z,Y) = H(X/Z)$, cioè quando $I(X \wedge Y/Z) = 0$, il che significa che Y non porta ulteriore informazione su X rispetto a quanta ne porti già Z. Questo potrebbe essere il caso di un rapporto funzionale (e biunivoco) tra le Y e Z.

Teorema 2.8 (II teorema di elaborazione dei dati). *Se* $X \to Y \to Z$ *allora*

$$I(X \wedge Z) \leq \begin{cases} I(X \wedge Y) \\ I(Y \wedge Z) \end{cases}$$

Dim. $I(X \wedge Z) = H(X) - H(X/Z) \leq H(X) - H(X/Y) = I(X \wedge Y)$, dove la disuguagianza segue dal I teorema di elaborazione dei dati. Dal fatto che anche $Z \to Y \to X$ segue $I(X \wedge Z) \leq I(Y \wedge Z)$ □

Anche questo secondo teorema può essere interpretato efficacemente affermando che due v.a. "lontane" si scambiano meno informazione di due variabili più "vicine". Come accennato fin dall'inizio i teoremi sono suscettibili di generalizzazioni a un numero qualunque di variabili. Per esempio se $X \to Y \to Z \to V$, dalle sottocatene $X \to Y \to Z$ e $X \to Z \to V$ si ricava $I(Y \wedge Z) \geq I(X \wedge Z) \geq I(X \wedge V)$, cioè le due variabili più interne si scambiano più informazione di quelle a loro più esterne. La dipendenza stocastica tra due variabile tende dunque a diventare sempre più rarefatta man mano che le due variabili si allontanano tra loro. L'estensione al caso di variabili vettoriali è immediata; nel seguito ci interesserà in particolare l'applicazione del II teorema di elaborazione dei dati alla catena $\boldsymbol{U}^{\boldsymbol{k}} \to \boldsymbol{X}^{\boldsymbol{n}} \to \boldsymbol{Y}^{\boldsymbol{n}} \to \boldsymbol{V}^{\boldsymbol{k}}$ (vedi paragrafo 4.4.1), che consente di scrivere

$$I(\boldsymbol{U}^{\boldsymbol{k}} \wedge \boldsymbol{V}^{\boldsymbol{k}}) \leq I(\boldsymbol{X}^{\boldsymbol{n}} \wedge \boldsymbol{Y}^{\boldsymbol{n}}) \tag{2.36}$$

Capitolo 3

Codifica di sorgente

3.1 L'entropia di una sorgente d'informazione

Entriamo ora nel vivo del modello del *sistema di comunicazione unidirezionale* descritto nella sezione 1.1, la cui definizione nel celebre articolo di Shannon [77] fu il punto di partenza per lo sviluppo successivo di tutta la teoria.

Nel modello c'è una *sorgente d'informazione* $\mathcal{S}$ la quale, in corrispondenza di ciascun istante di tempo (discreto) t_j $(\dots t_{-1}, t_0, t_1, t_2 \dots)$, emette una lettera x_{i_j} $(\dots, x_{i_{-1}}, x_{i_0}, x_{i_1}, x_{i_2} \dots)$ appartenente all'alfabeto $\mathcal{X} = \{x_1, x_2, \dots, x_K\}$ della sorgente. Il funzionamento della sorgente può essere rappresentato mediante una successione $\dots, X_{-1}, X_0, X_1, X_2 \dots$ di variabili aleatorie. Se non si fanno particolari ipotesi sul processo stocastico, per descrivere in termini probabilistici l'uscita della sorgente è necessario fornire una famiglia infinita di distribuzioni di probabilità del tipo

$$\Pr\{X_{j+1}, X_{j+2}, \dots, X_{j+n} = x_{i_1}, x_{i_2}, \dots, x_{i_n}\} \tag{3.1}$$

con $-\infty < j < +\infty,\ n = 1, 2, \dots,\ \ x_{i_m} \in \mathcal{X}$ e $1 \leq m \leq n$. In un quadro così generale un tale modello di funzionamento della sorgente è di fatto impraticabile. Tuttavia si possono introdurre alcune ipotesi restrittive, che se da un lato rendono la trattazione meno generale dall'altro consentono di ricavare importanti risultati normativi. La prima ipotesi che discuteremo è quella di *stazionarietà* del processo stocastico (nel nostro caso parleremo di stazionarietà della sorgente), che riflette una situazione in cui il meccanismo aleatorio che genera la successione delle v.a. ha un funzionamento invariante nel tempo. Ne consegue che la valutazione della probablità associata alla realizzazione di un certo vettore aleatorio *non* risente di traslazioni lungo l'asse temporale

$$\begin{aligned}\Pr\{X_{j+1}, X_{j+2}, \dots, X_{j+n} = x_{i_1}, x_{i_2}, \dots, x_{i_n}\} = \\ = \Pr\{X_{j+1+r}, X_{j+2+r}, \dots, X_{j+n+r} = x_{i_1}, x_{i_2}, \dots, x_{i_n}\}\end{aligned} \tag{3.2}$$

$r = 0, 1, 2, \ldots$. L'ipotesi di stazionarietà non è in pratica troppo restrittiva, e anche quando essa viene a mancare si può tentare una segmentazione dell'asse temporale in sotto-intervalli sostanzialmente stazionari.

Un'altra importante ipotesi che useremo nella nostra trattazione è *l'assenza di memoria* del processo. In questo caso c'è una indipendenza tra le variabili aleatorie della successione; in altre parole le realizzazioni passate non influenzano quelle future, e la probabilità del vettore aleatorio può essere espressa come prodotto delle probabilità delle singole componenti:

$$\Pr\{X_{j+1}, X_{j+2}, \ldots, X_{j+n} = x_{i_1}, x_{i_2}, \ldots, x_{i_n}\} = \prod_{r=1}^{n} \Pr\{X_{j+r} = x_{i_r}\} \quad (3.3)$$

La contemporanea validità delle ipotesi di *stazionarietà e assenza di memoria* consente di prescindere nella valutazione della (3.3) dalla posizione temporale associata all'indice j; ponendo p.es. $j = 0$ si ottiene

$$\Pr\{X_{j+1}, X_{j+2}, \ldots, X_{j+n} = x_{i_1}, x_{i_2}, \ldots, x_{i_n}\} = \prod_{r=1}^{n} \Pr\{X_r = x_{i_r}\} \quad (3.4)$$

Sotto tali ipotesi è possibile parlare della probabilità che la *sorgente stazionaria e senza memoria* (*SSM*) emetta una lettera x_{i_r}, e si può associare alle lettere dell'alfabeto K-ario $\mathcal{X} = \{x_1, x_2, \ldots, x_K\}$ una d.p. $P = \{p_1, p_2, \ldots, p_K\}$. In tal modo si ha la possibilità di definire *l'entropia della sorgente* $\mathcal{S}$

$$H(\mathcal{S}) = H(X) = -\sum_{i=1}^{K} p_i \log p_i \quad (3.5)$$

che esprime la *quantità media d'informazione emessa dalla sorgente* $\mathcal{S}$. Il funzionamento di una *SSM* può essere descritto anche mediante una sua *estensione* n-esima $\mathcal{S}^n$, nella quale vengono emesse n-ple di lettere prese dall'alfabeto $\mathcal{X}^n$. La d.p. che rimane associata al funzionamento della sorgente è determinata dalle ipotesi *SSM*, e risulta essere P^n. L'entropia della $\mathcal{S}^n$ si definisce in modo analogo alla (3.5)

$$\begin{aligned} H(\mathcal{S}^n) =& H(X_1, X_2, \ldots, X_n) = H(\boldsymbol{X^n}) = \\ =& -\sum_{\boldsymbol{x} \in \mathcal{X}^{\mathbf{n}}} p(\boldsymbol{x}) \log p(\boldsymbol{x}) = nH(\mathcal{S}) \end{aligned} \quad (3.6)$$

derivando l'ultima uguaglianza dalle ipotesi di assenza di memoria.

Se nel processo sono presenti fenomeni di memoria non è possibile sfruttare la (2.26), e dunque le definizioni (3.5) e (3.6) non sono definibili. In tal caso non è chiaro come si possa misurare l'informazione che pur esce dalla sorgente, e si pone addirittura il problema dell'esistenza di una definizione consistente per

l'entropia di una sorgente con memoria. Come vedremo subito è però sufficiente ipotizzare la sola stazionarietà per riuscire ad esprimere una definizione coerente. Per effetto di quest'ultima ipotesi possiamo far riferimento al vettore $X_1, X_2, \ldots, X_n$ prescindendo dalla sua collocazione sull'asse temporale. Seguendo lo spirito della (3.6) si può allora tentare una definizione di entropia per una sorgente stazionaria che si basi sul rapporto

$$H_n = H(X_1, X_2, \ldots, X_n)/n \tag{3.7}$$

quando $n \to \infty$. Un altro metodo altrettanto ragionevole è quello di valutare asintoticamente la quantità $h_n = H(X_n/X_1, \ldots, X_{n-1})$. Definiamo allora, se esistono, i seguenti due limiti

$$\begin{aligned} H(\mathcal{S}) &\triangleq \lim_{n\to\infty} \frac{H(X_1, X_2, \ldots, X_n)}{n} \\ h(\mathcal{S}) &\triangleq \lim_{n\to\infty} H(X_n/X_1, \ldots, X_{n-1}) \end{aligned} \tag{3.8}$$

Si osservi che a priori l'esistenza dei due limiti non è scontata. Ancor meno scontato è il fatto che essi coincidano.

Teorema 3.1. *Se la sorgente $\mathcal{S}$ è stazionaria valgono le seguenti proprietà:*

i) $h_{n-1} \geq h_n$
ii) $H_n \geq h_n$
iii) $H_{n-1} \geq H_n$
iv) $H(\mathcal{S})$ *e* $h(\mathcal{S})$ *esistono e coincidono*

Dim. La *i)* stabilisce che la successione $\{h_n\}$ è monotona non crescente; essendo tutti i suoi termini non negativi essa ammette limite. La dimostrazione è immediata

$$\begin{aligned} h_n =& H(X_n/X_1, X_2, \ldots, X_{n-1}) \leq H(X_n/X_2, \ldots, X_{n-1}) \\ =& H(X_{n-1}/X_1, \ldots, X_{n-2}) = h_{n-2} \end{aligned}$$

dove la penultima uguaglianza deriva dalla stazionarietà del processo. Per la *ii)* possiamo scrivere

$$\begin{aligned} H_n =& \frac{H(X_1, X_2, \ldots, X_n)}{n} = \\ =& \frac{1}{n}\sum_{i=1}^{n} H(X_i/X_1, X_2, \ldots, X_{i-1}) = \frac{1}{n}\sum_{i=1}^{n} h_i \geq h_n \end{aligned}$$

dove la seconda uguaglianza segue dalla regola della catena (2.23) e la disuguaglianza dalla *i)*. Si stabilisce così che la successione $\{H_n\}$ domina la successione $\{h_n\}$. Per la $iii)$ abbiamo invece

$$H_n = \frac{H(X_1, \ldots, X_n)}{n} = \frac{1}{n}\left[H(X_1, \ldots, X_{n-1}) + H(X_n/X_1, \ldots, X_{n-1})\right] \leq$$
$$\leq \frac{n-1}{n} H_{n-1} + \frac{1}{n} H_n$$

con la disuguaglianza che segue dalla *ii)*. Dal confronto tra gli estremi della catena segue infine la tesi. Dunque, anche la successione $\{H_n\}$ è monotona non crescente e ammette limite.

Per quanto riguarda la $iv)$ si può ricorrere al *Lemma di Cesáro* (se la successione h_n ha limite e converge a $h(\mathcal{S})$, anche la successione delle medie $H_n = 1/n \sum h_n$ converge a $h(\mathcal{S})$). Si può però procedere anche direttamente nel modo seguente. Dalla $ii)$ e dall'esistenza degli stessi si ricava subito

$$H(\mathcal{S}) \geq h(\mathcal{S}) \tag{3.9}$$

Per stabilirne l'uguaglianza sarebbe a questo punto sufficiente verificare anche la disuguaglianza opposta $H(\mathcal{S}) \leq h(\mathcal{S})$. Consideriamo a tal scopo un doppio indice nel processo al limite che porta a $H(\mathcal{S})$, valutando

$$H_{n+j} = \frac{1}{n+j} H(X_1, \ldots, X_{n-1}) + \frac{1}{n+j}\left[\sum_{i=n}^{n+j} H(X_i/X_1, \ldots, X_{i-1})\right] \leq$$
$$\leq \frac{n-1}{n+j} H_{n-1} + \frac{j+1}{n+j} h_n$$

dove la disuguaglianza segue dalla *i)*. Se ora imponiamo $j \to \infty$ si ottiene $H(\mathcal{S}) \leq h_n$; passando al limite per $n \to \infty$ si ottiene $H(\mathcal{S}) \leq h(\mathcal{S})$, che tenuto della (3.9), fornisce la tesi. □

La coincidenza dei due limiti ci autorizza a definire l'entropia di una sorgente stazionaria e senza memoria come limite asintotico dell'entropia vettoriale normalizzata o dell'entropia condizionata alla conoscenza di tutte le variabili precedenti.

3.2 Le leggi dei grandi numeri

Mettiamoci ora nell'ipotesi più restrittiva di stazionarietà e assenza di memoria. In questo caso la descrizione del funzionamento della sorgente può fare utile riferimento alle *leggi dei grandi numeri*, ben note in ambito probabilistico. Per comodità del lettore proponiamo un sommario delle proprietà più importanti. Se $\{X_n\} = X_1, X_2, \ldots, X_n, \ldots$ è la successione di v.a. *indipendenti e identicamente distribuite*, ci occuperemo del comportamento al limite della successione $\{1/n \cdot \sum_{i=1}^{n} X_i\}$ della media aritmetica delle v.a.

Def. 3.1 (Convergenza in probabilità). *Si dice che la successione* $\{X_n\}$ *converge in probabilità alla v.a.* X *se,* $\forall \epsilon, \delta > 0$, $\exists\, n_{\epsilon,\delta}$ *tale che,* $\forall n > n_{\epsilon,\delta}$ *risulta*

$$Pr\{|X_n - X| < \delta\} \geq 1 - \epsilon$$

La definizione implica un *numero infinito di affermazioni probabilistiche simultanee* su tutti i membri di una successione, (tranne che per un numero finito di essi, cioè i primi $n-1$). In altre parole si sta imponendo che sia

$$\Pr\{|X_n - X| < \delta\} \geq 1 - \epsilon, \quad \Pr\{|X_{n+1} - X| < \delta\} \geq 1 - \epsilon, \quad \Pr\{|X_{n+2} - X| < \delta\} \geq 1 - \epsilon, \ldots \tag{3.10}$$

Una scrittura equivalente per la definizione 3.1 è la seguente

$$\forall \delta > 0 \qquad \lim_{n\to\infty} \Pr\{|X_n - X| < \delta\} = 1 \tag{3.11}$$

o anche la

$$\lim_{n\to\infty} \text{ prob } X_n = X \tag{3.12}$$

Def. 3.2 (Convergenza quasi certa). *Si dice che la successione* $\{X_n\}$ *converge quasi certamente alla v.a.* X *se,* $\forall \epsilon, \delta > 0$, $\exists\, n_{\epsilon,\delta}$ *tale che,* $\forall n > n_{\epsilon,\delta}$ *e per ogni* $j > 0$ *risulta*

$$Pr\left\{\bigcap_{r=n}^{n+j} |X_r - X| < \delta\right\} \geq 1 - \epsilon$$

Si osservi la differenza rispetto alla definizione 3.1; in questo caso l'affermazione probabilistica è molto più forte, poichè si richiede la *contemporanea validità di una successione infinita di disuguaglianze*

$$\Pr\left\{ |X_n - X| < \delta,\ |X_{n+1} - X| < \delta,\ |X_{n+2} - X| < \delta, \ldots \right\} \geq 1 - \epsilon \tag{3.13}$$

che normalmente si esprime sinteticamente con la notazione

$$\Pr\left\{\lim_{n\to\infty} X_n = X\right\} = 1 \tag{3.14}$$

Assegnata la successione $\{X_1, X_2, \ldots, X_n\}$ (con medie finite) si dice allora che la stessa successione soddisfa la *legge dei grandi numeri* se

$$\frac{X_1 + X_2 + \ldots + X_n}{n} - \frac{E[X_1 + X_2 + \ldots + X_n]}{n} \longrightarrow 0 \tag{3.15}$$

per qualche tipo di convergenza quando $n \to \infty$. Se la convergenza è quella in probabilità si parlerà di *legge debole dei grandi numeri*, mentre per la convergenza quasi certa si introduce la locuzione di *legge forte dei grandi numeri*. Si osservi la differenza operativa nel comportamento della successione rispetto alle due modalità di convergenza, cioè essenzialmente rispetto alla (3.10) o alla (3.13). Nel primo caso si ha una elevata probabilità (pari a $1-\epsilon$) che X_n stia nella banda $X \pm \delta$ *per ogni singolo valore* n dell'ascissa temporale; ciò non esclude che per *qualche* valore di n ci si trovi al di fuori, anche se ciò succede con bassa probabilità. La (3.13) ci dice invece che *quasi tutte le sequenze* stanno all'interno della banda $X \pm \delta$; ovviamente nessuno esclude che la successione effettiva delle realizzazioni sia una di quelle che stanno (anche parzialmente) fuori dalla banda; ma queste sono poco frequenti, poiché cumulano una probabilità al più pari a ϵ.

Nel seguito ci occuperemo solo di convergenza in probabilità, e per tale motivo formalizziamo meglio la definizione di convergenza debole

Def. 3.3 (Legge debole dei grandi numeri). *Si dice che la successione* $\{X_n\}$ *soddisfa la legge debole dei grandi numeri se*

$$\lim_{n\to\infty} Pr\left\{\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \frac{1}{n}\sum_{i=1}^{n} E[X_i]\right| < \delta\right\} = 1 \tag{3.16}$$

Rimane ora da stabilire sotto quali condizioni una successione soddisfi la legge dei grandi numeri (nella versione debole o forte). Senza entrare nei dettagli (per una trattazione esauriente rimandiamo ai testi di Feller [34] o, in italiano, di Daboni [22]), possiamo solo accennare al fatto che di norma l'esistenza della media è sufficiente per la convergenza tanto debole che forte (teoremi di Khinčin e di Kolmogorov). Come già preannunciato, in questa sede ci occuperemo solo di convergenza debole, che può essere risolta sulla base dei celeberrimi risultati dovuti a Čebišev (l'omonima disuguaglianza e il teorema sulla convergenza). Data la loro importanza e tenendo conto che verranno usati in più occasioni nel corso della nostra trattazione riteniamo opportuno descriverli compiutamente.

Teorema 3.2 (Disuguaglianza di Čebišev). *Data una v.a.* X *di varianza* $Var[X] = E\left[(X - E[X])^2\right]$ *finita,* $\forall \delta > 0$ *vale la seguente disuguaglianza*

$$Pr\left\{|X - E[X]| \geq \delta\right\} \leq \frac{Var[X]}{\delta^2} \tag{3.17}$$

Dim.

$$\Pr\left\{|X - E[X]| \geq \delta\right\} = \sum_{x:|x-E[X]|\geq\delta} p(x) \leq \sum_{x:|x-E[X]|\geq\delta} \frac{(x-E[X])^2}{\delta^2} p(x) \leq$$
$$\leq \frac{1}{\delta^2} \sum_{\text{tutti gli } x} (x - E[X])^2 p(x) = \frac{Var[X]}{\delta^2}$$

dove la prima disuguaglianza deriva dal fatto che $1 \leq \frac{|x-E[X]|}{\delta}$ per gli x tali che $|x - E[X]| \geq \delta$, e la seconda dal fatto che la somma è a termini non negativi. □

La disuguaglianza di cui sopra è lo strumento chiave per la dimostrazione del teorema che fissa le condizioni per la validità della convergenza debole.

Teorema 3.3 (Teorema di Čebišev). *Data una successione $\{X_n\}$ di v.a. indipendenti, se la varianza delle variabili X_n è limitata, allora la successione soddisfa la legge debole dei grandi numeri.*

Dim. È noto che $Var[\alpha X + \beta] = \alpha^2 Var[X]$. Se $Var[X_n] \leq Cost$, per la limitatezza delle varianze e per l'indipendenza delle v.a. possiamo scrivere $Var\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n^2}\sum_{i=1}^{n} Var[X_i] \leq \frac{Cost}{n}$, e dunque anche la v.a. $\frac{1}{n}\sum_{i=1}^{n} X_i$ ha varianza limitata. Possiamo allora applicare la disuguaglianza di Čebišev (3.17)

$$\Pr\left\{\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \frac{1}{n}\sum_{i=1}^{n} E[X_i]\right| < \delta\right\} \geq 1 - \frac{Var\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right]}{\delta^2} \geq$$

$$\geq 1 - \frac{Cost}{n\delta^2} \tag{3.18}$$

cioè la tesi passando al limite per $n \to \infty$ □

La validità di questo teorema implica, come corollario, quella del noto teorema di Bernoulli per prove indipendenti .

Teorema 3.4 (Teorema di Bernoulli). *Se le v.a. della successione $\{X_n\}$ sono indipendenti e identicamente distribuite (i.i.d.) allora, detto $n(a)$ il numero occorrenze di un evento a di probabilità $p(a)$, si ha $\forall \delta > 0$*

$$\lim_{n\to\infty} Pr\left\{\left|\frac{n(a)}{n} - p(a)\right| < \delta\right\} = 1 \tag{3.19}$$

Si noti che, a norma delle osservazioni fatte precedentemente, la convergenza in senso debole (o forte) non implica che la frequenza relativa di un evento tende alla probabilità a priori, bensì che la *probabilità* di tale evenienza è elevata (nel senso della convergenza debole o in quello della convergenza forte).

3.3 Sequenze tipiche

3.3.1 Tipicità in probabilità

La legge debole dei grandi numeri può essere impiegata per descrivere il comportamento asintotico di una sorgente stazionaria e senza memoria, nel senso che da essa si ricavano informazioni sulla struttura asintotica delle sequenze

emesse dalla sorgente. Accanto a una formulazione in termini probabilistici esiste però anche una controparte riguardante lo studio della *cardinalità* delle sequenze che obbediscono alla legge debole. Mentre la prima impostazione è nata nel contesto della teoria delle probabilità e delle variabili aleatorie, la controparte in cardinalità della legge debole dei grandi numeri è stata sviluppata solo di recente (da Shannon, vedi [77] e da Csiszár e Körner; per una rassegna recente si veda invece [20] [21]).

Se consideriamo la successione $\{X_n\}$ di v.a. indipendenti e identicamente distribuite (cioè un processo stazionario e senza memoria), sappiamo che a norma della legge debole dei grandi numeri $\frac{1}{n}\sum_{i=1}^{n} X_i \stackrel{prob}{\rightarrow} E[X]$. Se applichiamo la stessa legge alla v.a. associata all'autoinformazione $\mathcal{I}(X) = -\log p(X)$ troviamo che

$$-\frac{1}{n}\sum_{i=1}^{n}\log p(X_i) = -\frac{1}{n}\log p(\boldsymbol{X^n}) = \frac{1}{n}\mathcal{I}(\boldsymbol{X^n}) \stackrel{prob}{\longrightarrow}$$

$$\stackrel{prob}{\longrightarrow} E[-\log p(X)] = -\sum_{j=1}^{K} p(x_j)\log p(x_j) = H(P)$$

con $P = \{p(x_i)\}_{i=1}^{K}$. In altre parole

$$\lim_{n\to\infty} \text{prob}\ \frac{1}{n}\mathcal{I}(\boldsymbol{X^n}) = H(P) \tag{3.20}$$

e dunque *l'autoinformazione normalizzata converge in probabilità all'entropia*. In termini probabilistici si può affermare che, a norma della (3.10), $\forall\delta > 0\ \exists\ \epsilon$, funzione di n e δ, tale che

$$\Pr\left\{\boldsymbol{x} : \left|\frac{1}{n}\mathcal{I}(\boldsymbol{x}) - H(P)\right| \leq \delta\right\} \geq 1 - \epsilon \tag{3.21}$$

Ciò stabilisce che asintoticamente *quasi tutte le sequenze che escono dalla sorgente hanno un'autoinformazione normalizzata prossima all'entropia*. Vale allora la pena di dare un nome specifico a queste sequenze:

Def. 3.4. *Una sequenza $\boldsymbol{x} \in \mathcal{X}^n$ si dice (n,δ)-tipica in probabilità se*

$$\left|\frac{1}{n}\mathcal{I}(\boldsymbol{x}) - H(P)\right| \leq \delta \tag{3.22}$$

Indichiamo con $\mathcal{T}_{\mathcal{P}}^{(n,\delta)}(P)$ (o più semplicemente con $\mathcal{T}_{\mathcal{P}}^{(n,\delta)}$ o $\mathcal{T}_p$) l'insieme delle sequenze tipiche in probabilità. La definizione 3.4 ha come immediata conseguenza il seguente teorema, che fissa la probabilità di una sequenza tipica e la cardinalità di $\mathcal{T}_p$.

Teorema 3.5. *Assegnati $\mathcal{T}_p^{(n,\delta)}$ e una qualunque sua sequenza $\boldsymbol{x}$ valgono le seguenti disuguaglianze*

$$2^{-n[H(P)+\delta]} \leq p(\boldsymbol{x}) \leq 2^{-n[H(P)-\delta]} \tag{3.23}$$

$$(1-\epsilon)\, 2^{n[H(P)-\delta]} \leq \left|\mathcal{T}_p^{(n,\delta)}\right| \leq 2^{n[H(P)+\delta]} \tag{3.24}$$

Dim. Dalla definizione di tipicità (3.22) si ricava

$$+\delta \geq -\frac{1}{n}\log p(\boldsymbol{x}) - H(P) \geq -\delta$$
$$-n[H(P)+\delta] \leq \log p(\boldsymbol{x}) \leq -n[H(P)-\delta]$$

cioè la (3.23). Per la cardinalità si ha invece

$$1 \geq \Pr\left\{\mathcal{T}_p^{(n,\delta)}\right\} \geq \left|\mathcal{T}_p^{(n,\delta)}\right| \min_{\boldsymbol{x}\in\mathcal{T}_p^{(n,\delta)}} p(\boldsymbol{x}) \geq \left|\mathcal{T}_p^{(n,\delta)}\right| 2^{-n[H(P)+\delta]}$$

e dal confronto tra gli estremi segue la limitazione superiore della (3.24). Per quella inferiore si ha invece

$$1-\epsilon \leq \Pr\left\{\mathcal{T}_p^{(n,\delta)}\right\} \leq \left|\mathcal{T}_p^{(n,\delta)}\right| \max_{\boldsymbol{x}\in\mathcal{T}_p^{(n,\delta)}} p(\boldsymbol{x}) \leq \left|\mathcal{T}_p^{(n,\delta)}\right| 2^{-n[H(P)-\delta]} \square$$

Il teorema stabilisce che ciascuna sequenza tipica ha una probabilità sostanzialmente uniforme e pari a (circa) $2^{-nH(P)}$. Poiché la probabilità dell'insieme $\mathcal{T}_p^{(n,\delta)}$ è quasi uno, è evidente che la cardinalità dello stesso dev'essere prossima a $2^{nH(P)}$, come confermato del resto dalla (3.24). Il rapporto tra cardinalità delle tipiche e cardinalità di tutte le sequenze è infinitesimo; infatti

$$\frac{\left|\mathcal{T}_p^{(n,\delta)}\right|}{K^n} \leq 2^{-n[\log K - H(P) - \delta]} \to 0 \tag{3.25}$$

quando $\log K > H(P) + \delta$. Ciò succede quasi sempre, poiché $\log K$ è una limitazione superiore per l'entropia che viene raggiunta solo nel caso di d.p. uniforme. Se ciò accade si ha $H(P) = \log K$, e *tutte* le sequenze diventano tipiche. Se definiamo

$$\mathcal{T}_p^{(n,\delta)} = \left\{\boldsymbol{x} : \left|\frac{1}{n}\mathcal{I}(\boldsymbol{x}) - H(P)\right| \leq \delta\right\} \tag{3.26}$$

si ha

$$\Pr\left\{\mathcal{T}_p^{(n,\delta)}\right\} \geq 1-\epsilon \qquad \text{con} \qquad \lim_{n\to\infty} \epsilon = 0$$

$$\left|\mathcal{T}_p^{(n,\delta)}\right| \approx 2^{nH(P)} \tag{3.27}$$

La (3.25) dimostra che asintoticamente le sequenze tipiche in probabilità hanno una cardinalità percentualmente infinitesima, a fronte del fatto che *cumulano su di sé tutta la probabilità*. In altre parole il comportamento asintotico di una *SSM* si può riassumere affermando che essa emette, con una probabilità prossima a uno, *solo* sequenze tipiche, che appartengono a un insieme (percentualmente) infinitesimo. Questo è il cosiddetto principio di *equipartizione asintotica* delle sequenze, che si può riformulare affermando anche che le sequenze in uscita dalla sorgente si dividono in due sottinsiemi:

1. quello delle sequenze tipiche, a probabilità sostanzialmente uniforme, che sono in numero di $2^{nH(P)}$ e la cui cardinalità è infinitesima se rapportata al numero totale; per queste sequenze la probabilità cumulata è prossima a uno;

2. l'insieme complementare, che contiene tutte le altre sequenze (che sono $K^n - 2^{nH(P)}$, cioè quasi tutte quelle possibili), la cui probabilità asintotica è però nulla.

Il principio di equipartizione asintotica non vale solo per sorgenti stazionarie e senza memoria, ma anche per la classe più generale delle sorgenti *ergodiche*. Una definizione generale di processo ergodico coinvolgerebbe nozioni di teoria della misura, che esulano le finalità di questa trattazione. Ci limiteremo a fornire una definizione più intuitiva, che consiste nel caratterizzare l'ergodicità come quella proprietà grazie alla quale le medie temporali coincidono, nel senso delle leggi dei grandi numeri, con le medie d'insieme.

Assegnata dunque una sorgente $\mathcal{S}$ sull'alfabeto $\mathcal{X} = \{x_1, x_2, \ldots, x_K\}$ sia $\boldsymbol{x} = x_{i_1} x_{i_2} \ldots x_{i_n}$ la sequenza di lunghezza n generata dalla sorgente nei primi n istanti di tempo. Consideriamo ora una stringa $\boldsymbol{\alpha} = \alpha_{j_1} \alpha_{j_2} \ldots \alpha_{j_m}$, con $\alpha_{j_r} \in \mathcal{X}$, $(1 \leq r \leq m)$, e definiamo con $N_n(\boldsymbol{\alpha}, \boldsymbol{x})$ il numero di volte in cui la stringa $\boldsymbol{\alpha}$ compare nella sequenza $\boldsymbol{x}$ di lunghezza n.

Def. 3.5. *Si dice che una sorgente è ergodica se essa è stazionaria e se* $\forall \delta$ *e per qualsiasi sequenza* $\boldsymbol{\alpha}$ *accade che*

$$\lim_{n \to \infty} \Pr \left\{ \left| \frac{N_n(\boldsymbol{\alpha}, \boldsymbol{x})}{n} - \Pr \{X_1, \ldots, X_m = \alpha_{j_1} \ldots \alpha_{j_m}\} \right| < \delta \right\} = 1$$

cioè $N_n(\boldsymbol{\alpha}, \boldsymbol{x})/n$ *converge in probabilità a* $\Pr \{X_1, \ldots, X_m = \alpha_{j_1} \ldots \alpha_{j_m}\}$.

I risultati ottenuti in questa sezione, e riguardanti essenzialmente la cardinalità delle sequenze che obbediscono alla legge debole dei grandi numeri, saranno sfruttati nel seguito per effettuare una compressione dei dati. Possiamo anticipare l'idea seguente: poiché la sorgente emette asintoticamente solo "poche" sequenze tra tutte le possibili, cioè le tipiche in probabilità, è sufficiente codificare

solo queste ultime, realizzando così una *compressione* dei dati. L'ergodicità ci consentirà inoltre di trattare le medie d'insieme come se fossero medie temporali, consentendoci di stabilire delle strategie per rendere minime le sequenze secondarie che escono dal decodificatore lavorando però sulla distribuzione di probabilità della sorgente. Ma di ciò parleremo più diffusamente nella sezione 3.5.

La proprietà di equipartizione asintotica si può generalizzare anche al caso di sorgenti con memoria e, dal punto di vista operativo, anche alle sorgenti che producono un *linguaggio naturale*. In quest'ultimo caso la proprietà di equipartizione asintotica distingue tra *sequenze significative* (cioè frasi di senso compiuto nel linguaggio naturale) e *sequenze non significative*, cioè prive di senso. Di ciò parleremo però più diffusamente nella terza sezione dedicata ai cifrari.

Esempio 3.1. Sia $n = 100$, $K = 2$, $P = \{p_0, p_1\}$ con $p_0 = 0.89$, $p_1 = 0.11$, cui corrisponde $h(P) = 0.5$. Fissiamo inoltre $\delta = 0.061$. L'insieme $\mathcal{T}_{\mathcal{P}}^{(100,0.061)}$ delle sequenze $(100, 0.061)$-tipiche è costituito tutte quelle sequenze che soddisfano la relazione (3.22)

$$\left| \frac{1}{100} \mathcal{I}(\boldsymbol{x}) - 0.5 \right| \leq 0.061$$

Sulla base della (3.23) possiamo scrivere

$$\begin{aligned} 2^{-100[0.5+0.061]} \leq p(\boldsymbol{x}) \leq 2^{-100[0.5-0.061]} \qquad & \text{cioè} \\ 1.295\, 10^{-17} \leq p(\boldsymbol{x}) \leq 6.029\, 10^{-14} \qquad & (3.28) \end{aligned}$$

Dalla tabella che segue si può verificare che le tipiche sono tutte le sequenze di peso 9, 10, 11, 12, e 13, le uniche a soddisfare la (3.28). Infatti

Peso	$p(\boldsymbol{x})$	Cardinalità	Prob. tot.
9	$0.89^{91}\, 0.11^{9} = 5.848\, 10^{-14}$	$\binom{100}{9} = 1.902\, 10^{12}$	0.111
10	$0.89^{90}\, 0.11^{10} = 7.228\, 10^{-15}$	$\binom{100}{10} = 1.731\, 10^{13}$	0.125
11	$0.89^{89}\, 0.11^{11} = 8.943\, 10^{-16}$	$\binom{100}{11} = 1.416\, 10^{14}$	0.126
12	$0.89^{88}\, 0.11^{12} = 1.104\, 10^{-16}$	$\binom{100}{12} = 1.050\, 10^{15}$	0.116
13	$0.89^{87}\, 0.11^{13} = 1.365\, 10^{-17}$	$\binom{100}{13} = 7.110\, 10^{15}$	0.097
Totale		$8.32\, 10^{15}$	0.576

Dalla tabella si ricava

$$\Pr\left\{\mathcal{T}_{\mathcal{P}}^{(100,0.061)}\right\} = 0.576 \qquad \left|\mathcal{T}_{\mathcal{P}}^{(100,0.061)}\right| = 8.32\, 10^{15}$$

e deve dunque essere $\epsilon \geq 1 - 0.576 = 0.424$ (in realtà si dovrebbe fissare prima ϵ e poi trovare n che soddisfa le relazioni). La cardinalità che abbiamo trovato soddisfa la limitazione (3.24)

$$(1-\epsilon)\, 2^{100[0.5-0.061]} \leq \left|\mathcal{T}_{\mathcal{P}}^{(100,0.061)}\right| \leq 2^{100[0.5+0.061]} \qquad \text{cioè}$$

$$9.45\, 10^{12} \leq \left|\mathcal{T}_{\mathcal{P}}^{(100,0.061)}\right| \leq 7.72\, 10^{16}$$

Poiché $2^{100} = 1.267\, 10^{30}$ la frazione di sequenze tipiche risulta pari a

$$\frac{8.32\, 10^{15}}{1.267\, 10^{30}} = 6.56\, 10^{-15}$$

e dunque una frazione pari a $6.56\, 10^{-15}$ delle sequenze totali cumula una probabilità pari a 0.576! ◯

3.3.2 Tipicità in composizione

Le sequenze tipiche in probabilità sono caratterizzate dal fatto che la loro autoinformazione normalizzata coincide essenzialmente con l'entropia. D'altra parte, quando osserviamo una sequenza sufficientemente lunga ci aspettiamo che anche la frequenza relativa di una singola lettera sia prossima alla probabilità a priori della stessa (sempre nel senso della legge debole dei grandi numeri). Di conseguenza, per effetto del teorema di Bernoulli , se n_i è l'occorrenza della i-esima lettera in una stringa di lunghezza n il rapporto $f_i = n_i/n$ tende in probabilità alla probabilità a priori p_i. La *composizione* delle sequenze in uscita, cioè la distribuzione n_i delle occorrenze di tutte le lettere, dovrebbe allora rispecchiare la composizione np_i indotta dal vettore di probabilità. Studieremo ora gli insiemi di sequenze che soddisfano questa condizione, tanto dal punto di vista della loro probabilità quanto, soprattutto, da quello della cardinalità.

Tipi esatti

Con le stesse notazioni della sezione precedente sia assegnata una distribuzione di probabilità $P = (p_1, p_2, \ldots, p_K)$ per le realizzazioni di una successione di v.a. i.i.d. Consideriamo una sequenza $\boldsymbol{x} = x_{i_1} x_{i_2} \ldots x_{i_n}$ di lunghezza n in uscita e facciamo riferimento alla sua composizione, cioè al vettore $(n_1, n_2, \ldots, n_K)$, dove con $n_i = N(x_i/\boldsymbol{x})$ indichiamo il numero di volte in cui la lettera x_i compare in $\boldsymbol{x}$. È ovvio che valgono i seguenti due vincoli

$$0 \leq n_i \leq n \qquad\qquad \sum_{i=1}^{K} n_i = n \tag{3.29}$$

Introduciamo ora la seguente

Def. 3.6. *Una sequenza $\boldsymbol{x}$ si dice tipica in composizione rispetto alla d.p. P del processo i.i.d. e alla costante δ se, $\forall i : 1 \leq i \leq K$ risulta*

$$\left|\frac{n_i}{n} - p_i\right| \leq \delta$$

(se $p_i = 0$ si pretende $n_i = 0$). Indichiamo con $\mathcal{T}_c^{(n,\delta)}(P)$ (o più semplicemente con $\mathcal{T}_c^{(n,\delta)}$ o con $\mathcal{T}_c$) l'insieme delle sequenze tipiche in composizione. Gran parte delle considerazioni che faremo nel seguito avranno un carattere asintotico; in particolare fisseremo al posto di δ una successione $\{\delta_n\}$ tale che

$$\lim_{n\to\infty} \delta_n = 0 \quad \text{ma} \quad \lim_{n\to\infty} \delta_n \sqrt{n} = +\infty \tag{3.30}$$

che corrisponde a imporre una valutazione sempre più severa, via via che n cresce, per il possesso del requisito di tipicità in composizione. Al limite si richiede una composizione perfettamente aderente al vettore P, anche se ci si accontenta di una velocità di convergenza molto modesta, come specificato dal secondo limite della (3.30); il motivo di ciò verrà compreso tra breve.

Intuitivamente si comprende che la richiesta di tipicità in composizione è più forte di quella di tipicità in probabilità; ciò viene confermato dal seguente

Teorema 3.6. *Se l'n-pla $\boldsymbol{x}$ è tipica in composizione essa è tipica anche in probabilità; cioè $\mathcal{T}_c^{(n,\delta)} \subset \mathcal{T}_p^{(n,\alpha)}$ con $\alpha = \delta \sum_{i=1}^{K}(-\log p_i)$.*

Dim. Per l'ipotesi i.i.d del processo possiamo scrivere

$$p(\boldsymbol{x}) = \prod_{i=1}^{K} p_i^{n_i} = 2^{\sum_{i=1}^{K} n_i \log p_i} \qquad \text{e dunque}$$

$$-\frac{1}{n}\log p(\boldsymbol{x}) - H(P) = -\frac{1}{n}\sum_{i=1}^{K} n_i \log p_i + \sum_{i=1}^{K} p_i \log p_i =$$

$$= \sum_{i=1}^{K}\left(\frac{n_i}{n} - p_i\right)(-\log p_i) \tag{3.31}$$

Ma per l'ipotesi di tipicità in composizione si ha $-\delta \leq \frac{n_i}{n} - p_i \leq \delta$, cioè

$$-\delta \sum_{i=1}^{K}(-\log p_i) \leq -\frac{1}{n}\log p(\boldsymbol{x}) - H(P) \leq \delta \sum_{i=1}^{K}(-\log p_i)$$

Essendo i p_i costanti e predeterminati, il termine $\alpha = \delta \sum_{i=1}^{K}(-\log p_i)$ è costante e stabilisce la (n, α) tipicità in probabilità della $\boldsymbol{x}$. □

Ci possono essere dunque delle sequenze che sono tipiche in probabilità, ma

non in composizione. Ciò è giustificato dal fatto che per la tipicità in probabilità ci si accontenta di avere una probabilità del vettore $\boldsymbol{x}$ che grossomodo corrisponda a quella tipica, anche se essa deriva da una composizione squinternata del vettore; p.es. un aumento di un n_i che corrisponda a una lettera molto probabile, e che faccia perdere la tipicità, può essere compensato dal punto di vista della probabilità globale del vettore inserendo qualche lettera in più tra quelle meno probabili (a spese di qualche altra più probabile).

Rimane ora da stabilire se anche l'insieme $\mathcal{T}_c^{(n,\delta)}$ saturi o meno asintoticamente tutta la probabilità diponibile.

Teorema 3.7. *Assegnato l'alfabeto $\mathcal{X}$, con $|\mathcal{X}| = K$, e una successione di v.a. i.i.d. con d.p. P, sia $\{\delta_n\}$ una successione tale che $\lim_{n\to\infty} \delta_n = 0$, ma con $\lim_{n\to\infty} \delta_n\sqrt{n} = \infty$; esiste allora una successione $\{\epsilon_n\}$, con $\lim_{n\to\infty} \epsilon_n = 0$, e che dipende solo da δ_n e K, tale che*

$$P^n(\mathcal{T}_c^{(n,\delta)}) \geq 1 - \epsilon_n$$

Dim. L'idea è quella di applicare la disuguaglianza di Čebišev (3.17) al vettore di lunghezza n. Per ogni $x_i \in \mathcal{X}$ consideriamo la v.a. N_i, che denota il numero di occorrenze della lettera x_i nel vettore $\boldsymbol{x}$, le cui componenti sono realizzazioni di v.a. i.i.d. secondo la d.p. P. Sia ora $\Delta_i(j)$ una v.a. binaria che assumendo il valore 1 specifica se in posizione j è uscita la lettera x_i ($\Delta_i(j) = 1$). Possiamo scrivere $N_i = \sum_{j=1}^{n} \Delta_i(j)$. Tenendo conto che il processo è Bernoulliano si ha $E[\Delta_i(j)] = p_i$ e $Var[\Delta_i(j)] = p_i(1-p_i)$ (si veda [34]). Dunque $E[N_i] = np_i$ e $Var[N_i] = np_i(1-p_i)$. Applichiamo ora la disuguaglianza di Čebišev

$$\Pr\{\boldsymbol{x} : |N_i - np_i| \geq n\delta_n\} \leq \frac{np_i(1-p_i)}{n^2\delta_n^2} \quad \text{cioè}$$
$$\Pr\left\{\boldsymbol{x} : \left|\frac{1}{n}N_i - p_i\right| \geq \delta_n\right\} \leq \frac{p_i(1-p_i)}{n\delta_n^2} \leq \frac{1}{4n\delta_n^2}$$

poiché $p_i(1-p_i) \leq \frac{1}{4}$. Ma

$$\Pr\left\{\boldsymbol{x} \notin \mathcal{T}_c^{(n,\delta)}\right\} = \Pr\left\{\bigcup_{i=1}^{K} \left|\frac{1}{n}N_i - p_i\right| \geq \delta_n\right\} \leq$$
$$\leq \sum_{i=1}^{K} \Pr\left\{\left|\frac{1}{n}N_i - p_i\right| \geq \delta_n\right\} \leq \frac{K}{4n\delta_n^2}$$

Dunque $\quad \Pr\left\{\boldsymbol{x} \in \mathcal{T}_c^{(n,\delta)}\right\} \geq 1 - \frac{K}{4n\delta_n^2}$

cioè la tesi ponendo $\epsilon_n = 1 - \frac{K}{4n\delta_n^2}$ □

Anche l'insieme delle sequenze tipiche in composizione, pur essendo un sottoinsieme di quelle tipiche in probabilità, cumula dunque una probabilità che si approssima asintoticamente a 1. Per quanto riguarda invece la cardinalità di $\mathcal{T}_c^{(n,\delta)}$ c'è da osservare che la sua valutazione è molto più onerosa del caso delle sequenze tipiche in probabilità. Si tratta infatti di mettere nell'insieme solamente quelle sequenze caratterizzate da tutte le possibili n-ple del tipo $n_1, n_2, \ldots, n_K$ che soddisfano la definizione 3.6 sotto le condizioni (3.29). Per specificare meglio il problema introduciamo una *relazione di equivalenza* in cui si definiscono equivalenti due n-ple che sono l'una permutazione dell'altra. La classe di equivalenza che rimane definita si chiama *tipo esatto* o, più semplicemente, *tipo*. Siamo subito in grado di calcolare la cardinalità del tipo esatto $T_{\boldsymbol{x}}$, la cui composizione è $(n_1, n_2, \ldots, n_K)$, e di cui $\boldsymbol{x}$ è un rappresentante

$$|T_{\boldsymbol{x}}| = \frac{n!}{n_1!\, n_2! \ldots n_K!} = \binom{n}{n_1\, n_2 \ldots n_K} \tag{3.32}$$

non essendo distinguibili tutte le permutazioni nelle quali si scambiano lettere uguali. Per quanto riguarda il numero di tipi possibili bisogna contare in quanti modi si può costruire una K-pla di interi $(n_1, n_2, \ldots, n_K)$ assoggettata ai vincoli (3.29). Un metodo semplice per impostare il calcolo è quello di associare a ciascuna unità di un generico n_j un simbolo grafico (p.es. un asterisco); se dunque $n_j = 3$ useremo per n_j tre asterischi. Poiché l'aumento (o la diminuzione) di n_j viene fatta sempre a scapito degli altri n_i, per rappresentare un tipo possiamo usare lo schema sotto riportato, caratterizzato da n asterischi che rappresentano le numerosità degli n_j e $K-1$ barre che delimitano le pertinenze degli stessi n_j (vedi il problema del riempimento delle urne [34])

$$\overbrace{*\,*\ldots*}^{n_1}\big|\overbrace{*\,*\ldots*}^{n_2}\big|\ldots\big|\overbrace{*\,*\ldots*}^{n_K}$$

Per esempio lo scambio della prima barra con l'asterisco alla sua destra comporta l'aumento di un'unità per n_1 e la diminuzione della stessa quantità per n_2. Di conseguenza ogni permutazione possibile degli $n + K - 1$ oggetti (barre e asterischi) descrive una diversa composizione per il tipo. Il numero totale Γ_n di tipi è dunque pari a

$$\Gamma_n = \frac{(n+K-1)!}{n!(K-1)!} = \binom{n+K-1}{n} \tag{3.33}$$

In molti ragionamenti asintotici che faremo, piuttosto che trattare direttamente col coefficiente binomiale che definisce Γ_n, risulta più agevole avere a disposizione una sua limitazione superiore, che si ottiene non appena si pensi che Γ_n

è il numero di K-ple di interi in $[0 \dots n]$ che sommano a n; togliendo quest'ultimo vincolo (il fatto che sommino a n) abbiamo una quantità pari a $(n+1)^K$. Ne segue che

$$\Gamma_n \leq (n+1)^K \tag{3.34}$$

Dunque le K^n sequenze si ripartiscono in "pochi" tipi, poiché $(n+1)^K$ cresce "solo" in modo polinomiale.

Tutto il resto della sezione lo dedicheremo allo studio della probabilità di un tipo esatto $T_{\boldsymbol{x}}$ e alla valutazione di un'approssimazione asintotica per la sua cardinalità $|T_{\boldsymbol{x}}|$, che possa essere impiegata al posto del coefficiente multinomiale (3.32).

Probabilità e cardinalità di un tipo esatto

Sia T_F un tipo esatto, $(n_1, n_2, \dots, n_K)$ la sua composizione, $\boldsymbol{x}_F$ un vettore rappresentante del tipo T_F ($\boldsymbol{x}_F \in T_F$) e $F = (f_1, f_2, \dots, f_K)$, $f_i = n_i/n$, il *vettore empirico di probabilità* associato al tipo T_F. La legge dei grandi numeri stabilisce che F sarà molto prossimo in probabilità al vettore P. La probabilità cumulata dal tipo T_F è pari alla somma delle probabilità di tutti i vettori che appartengono al tipo

$$P^n(T_F) = \sum_{\boldsymbol{y} \in T_F} P^n(\boldsymbol{y}) = P^n(\boldsymbol{x}) \cdot |T_F| \tag{3.35}$$

L'ultima uguaglianza deriva dal fatto che ogni tipo contiene solo n-ple che sono l'una permutazione dell'altra; valendo le ipotesi di stazionarietà e assenza di memoria si ricava allora che *tutti i vettori appartenenti allo stesso tipo hanno la stessa probabilità* $P^n(\boldsymbol{x})$. Per quanto riguarda la probabilità di un qualunque elemento $\boldsymbol{x}$ di T_F, sempre per le ipotesi di processo i.i.d. possiamo scrivere

$$\begin{aligned} P^n(\boldsymbol{x}) = p_1^{n_1} p_2^{n_2} \dots p_K^{n_K} &= 2^{\sum_{i=1}^K n_i \log p_i} = 2^{n \sum_{i=1}^K \frac{n_i}{n} \log p_i} = \\ &= 2^{-n \sum_{i=1}^K f_i \log \frac{1}{p_i}} = 2^{-n\left(\sum_{i=1}^K f_i \log \frac{1}{p_i} + \sum_{i=1}^K f_i \log f_i - \sum_{i=1}^K f_i \log f_i\right)} = \\ &= 2^{-n(H(F)+D(F/\!/P))} \end{aligned} \tag{3.36}$$

La relazione mostra che le n-ple con un vettore empirico F molto difforme da P hanno una probabilità molto bassa. Se in particolare facciamo funzionare il processo sulla base di una d.p. che coincide con quella del vettore empirico, cioè imponiamo $P \equiv F$, la divergenza si annulla, e troviamo subito una limitazione superiore per la cardinalità di un tipo esatto

$$F^n(T_F) = 2^{-nH(F)} \cdot |T_F| \leq 1 \tag{3.37}$$

cioè

$$|T_F| \leq 2^{nH(F)} \tag{3.38}$$

Tutto ciò è giustificato dal fatto che la cardinalità di un tipo non dipende dalla d.p. d'ingresso, ma solo dalla sua composizione; per calcolare una limitazione per la cardinalità possiamo allora usare la d.p. che più ci fa comodo.

A questo punto è necessario trovare una limitazione inferiore per verificare se quella superiore è sufficientemente stretta. Prima abbiamo però bisogno di un paio di risultati preparatori

Lemma 3.1. *Assegnati $n, m \in \mathbb{N}$ vale la relazione*

$$\frac{n!}{m!} \leq n^{n-m} \tag{3.39}$$

Dim. Se $n \geq m$ sappiamo che $n! = 1 \cdot 2 \cdot \ldots \cdot m \cdot (m+1) \cdot \ldots \cdot n$ e dunque

$$\frac{n!}{m!} = \frac{1 \cdot 2 \cdot \ldots \cdot m \cdot (m+1) \cdot \ldots \cdot n}{1 \cdot 2 \cdot \ldots \cdot m} = \overbrace{(m+1) \cdot \ldots \cdot n}^{n-m} \leq n^{n-m}$$

Se invece $n \leq m$ si può procedere in modo analogo

$$\frac{n!}{m!} = \frac{1 \cdot 2 \cdot \ldots \cdot n}{1 \cdot 2 \cdot \ldots \cdot n \cdot (n+1) \cdot \ldots \cdot m} = \frac{\overbrace{1}^{m-n}}{(n+1) \cdot \ldots \cdot m} \leq \frac{1}{n^{m-n}}$$

□

Indichiamo con T_G un qualunque altro tipo diverso da T_F.

Lemma 3.2. *Il tipo T_F ha la massima F-probabilità rispetto a qualunque altro tipo T_G, cioè*

$$F^n(T_F) \geq F^n(T_G) \tag{3.40}$$

Dim. Se $G = (g_1, g_2, \ldots, g_K)$ è il vettore empirico del tipo T_G, tenendo conto della (3.35) possiamo scrivere

$$F^n(T_F) = |T_F| \, f_1^{nf_1} f_2^{nf_2} \ldots f_K^{nf_K}$$
$$F^n(T_G) = |T_G| \, f_1^{ng_1} f_2^{ng_2} \ldots f_K^{ng_K}$$

Calcoliamo il rapporto

$$\frac{F^n(T_G)}{F^n(T_F)} = \frac{|T_G|\, f_1^{ng_1} f_2^{ng_2} \dots f_K^{ng_K}}{|T_F|\, f_1^{nf_1} f_2^{nf_2} \dots f_K^{nf_K}} =$$
$$= \frac{\binom{n}{ng_1\, ng_2 \dots ng_K}}{\binom{n}{nf_1\, nf_2 \dots nf_K}} \cdot f_1^{n(g_1-f_1)} f_2^{n(g_2-f_2)} \dots f_K^{n(g_K-f_K)} =$$
$$= \prod_{i=1}^{K} \frac{(nf_i)!}{(ng_i)!} f_i^{n(g_i-f_i)} \le \prod_{i=1}^{K} (nf_i)^{n(f_i-g_i)} f_i^{n(g_i-f_i)} =$$
$$= \prod_{i=1}^{K} n^{n(f_i-g_i)} = n^{n(\sum_{i=1}^{K} f_i)} \cdot n^{-n(\sum_{i=1}^{K} g_i)} = 1 \tag{3.41}$$

cioè la tesi, tenuto conto del lemma 3.1). □

Dal lemma 3.2 si può ricavare subito la limitazione inferiore cercata; usiamo come d.p. della sorgente la F

$$1 = \sum_{i=1}^{\Gamma_n} F^n(T_i) \le \Gamma_n\, F^n(T_F) = \Gamma_n F^n(\boldsymbol{x})\, |T_F| =$$
$$= \Gamma_n\, 2^{-nH(F)}\, |T_F| \le (n+1)^K 2^{-nH(F)}\, |T_F| \tag{3.42}$$

dove la prima disuguaglianza segue dalla (3.40) e la seconda dalla (3.34). Confrontando gli estremi della catena e tenendo conto della (3.38) possiamo fissare una limitazione inferiore e una superiore per la cardinalità del tipo esatto T_F

$$\frac{1}{(n+1)^K} 2^{nH(F)} \le |T_F| \le 2^{nH(F)} \tag{3.43}$$

Portando a esponente $(n+1)^K$ si deduce l'espressione

$$2^{n\left[H(F) - \frac{K}{n}\log(n+1)\right]} \le |T_F| \le 2^{nH(F)} \tag{3.44}$$

la cui versione asintotica, tenendo conto che $\frac{K}{n}\log(n+1) \to 0$ con $n \to \infty$, risulta essere

$$2^{n[H(F)-\epsilon_n]} \le |T_F| \le 2^{nH(F)} \tag{3.45}$$

con $\epsilon_n \to 0$ quando n. La limitazione superiore è dunque stretta e il coefficiente multinomiale, che esprime la cardinalità di un tipo esatto, può essere approssimato asintoticamente mediante il termine $2^{nH(F)}$

$$|T_F| = \binom{n}{n_1\, n_2 \dots n_K} \approx 2^{nH(F)} \tag{3.46}$$

L'entropia acquisisce dunque un'altra interpretazione operativa, in questo caso nel contesto della matematica combinatoria, come *approssimazione asintotica per il tasso di un coefficiente multinomiale*. In particolare, per il coefficiente binomiale possiamo scrivere

$$\binom{n}{i} = \binom{n}{i \quad n-i} \approx 2^{nh(\frac{i}{n})} \tag{3.47}$$

dove $h(p) = -p\log p - (1-p)\log(1-p)$ è l'entropia binaria.

Dalla (3.43) si ricava anche una limitazione per la probabilità di un tipo esatto, come specificato dal seguente

Teorema 3.8. *Se P è la d.p. associata a un processo i.i.d., la P-probabilità di un tipo T_F rimane limitata nel modo seguente*

$$\frac{1}{(n+1)^K} 2^{-nD(F/\!/P)} \leq P^n(T_F) \leq 2^{-nD(F/\!/P)} \tag{3.48}$$

Dim. Moltiplicando tutti i termini della (3.43) per la probabilità (3.36) di un vettore appartenente al tipo T_F si ricava

$$\begin{aligned}\frac{1}{(n+1)^K}\, 2^{nH(F)}\, 2^{-n(H(F)+D(F/\!/P))} \leq |T_F|\, P^n(\boldsymbol{x}_F) = \\ = P^n(T_F) \leq 2^{nH(F)}\, 2^{-n(H(F)+D(F/\!/P))}\end{aligned}$$

che è la tesi □

Siamo ora in grado di valutare la cardinalità dell'insieme $\mathcal{T}_C^{(n,\delta)}$ delle sequenze tipiche in composizione. Esso è costituito dall'unione di tutti i tipi esatti per le sequenze dei quali vale la definizione 3.6

$$\left|\mathcal{T}_C^{(n,\delta)}\right| = \left|\bigcup_{i\in\mathcal{I}_C} T_i\right| = \sum_{i\in\mathcal{I}_C} |T_i| \tag{3.49}$$

dove $\mathcal{I}_C$ è l'insieme degli indici relativi ai tipi che stanno nell'insieme. Si noti che ciascun tipo o sta interamente nell'insieme $\mathcal{T}_C^{(n,\delta)}$ o non vi sta con alcuno dei propri elementi. In altre parole $\mathcal{T}_C^{(n,\delta)}$ è *un'unione di tipi esatti* (tra loro disgiunti). Per trovare una limitazione inferiore e una superiore per la cardinalità, sulla base della (3.43) possiamo scrivere

$$(n+1)^{-K} \sum_{i\in\mathcal{I}_C} 2^{nH(F_i)} \leq \left|\mathcal{T}_C^{(n,\delta)}\right| = \sum_{i\in\mathcal{I}_C} |T_i| \leq \sum_{i\in\mathcal{I}_C} 2^{nH(F_i)} \tag{3.50}$$

Ma il numero di tipi esatti è limitato dalla (3.34), cioè $\Gamma_n \leq (n+1)^K$; possiamo allora limitare superiormente la (3.50), estendendo la sommatoria a tutti i tipi e

scegliendo, fra tutti i possibili, quello cui corrisponde la massima entropia. Per la limitazione inferiore possiamo invece trattenere, tra tutti i tipi della sommatoria, *solo* quello cui corrisponde l'entropia massima. Così facendo si ottiene

$$(n+1)^{-K} 2^{n \cdot \max_{i \in \mathcal{I}_C}\{H(F_i)\}} \leq \left|\mathcal{T}_C^{(n,\delta)}\right| \leq (n+1)^K 2^{n \cdot \max_{i \in \mathcal{I}_C}\{H(F_i)\}} \tag{3.51}$$

D'altra parte, poiché le sequenze sono tutte tipiche si ha $\left|f_i - p_i\right| \leq \delta_n \ \forall i$, con $\delta_n \to 0$ per effetto dell'ipotesi (3.30); possiamo allora scrivere

$$\left|\max_{i \in \mathcal{I}_C} H(F_i) - H(P)\right| \leq \Delta_n$$

con $\Delta_n > 0$ opportuno e tale che $\lim_{n\to\infty} \Delta_n = 0$. Anche per l'insieme delle sequenze tipiche in composizione vale allora una relazione simile alla (3.24)

$$2^{n\left[-\frac{K}{n}\log(n+1)+H(P)-\Delta_n\right]} \leq \left|\mathcal{T}_C^{(n,\delta)}\right| \leq 2^{n\left[\frac{K}{n}\log(n+1)+H(P)+\Delta_n\right]}$$

Ponendo $\alpha_n = \frac{K}{n}\log(n+1) + \Delta_n$ come correzione infinitesima possiamo scrivere una relazione analoga alla (3.24)

$$2^{n[H(P)-\alpha_n]} \leq \left|\mathcal{T}_C^{(n,\delta)}\right| \leq 2^{n[H(P)+\alpha_n]} \tag{3.52}$$

Il teorema 3.6 visto precedentemente anticipa allora il seguente risultato

$$\frac{\left|\mathcal{T}_C^{(n,\delta)}\right|}{K^n} \leq 2^{-n[\log K - H(P) - \alpha_n]} \to 0 \tag{3.53}$$

Anche le sequenze tipiche in composizione sono caratterizzate da una cardinalità percentualmente infinitesima, pur cumulando una probabilità che satura a 1 quando $n \to \infty$. Si potrebbe dimostrare che esse sono *il minimo* insieme che soddisfa queste condizioni (si veda [21]).

3.4 Codifica a lunghezza variabile

Nel seguito del capitolo manterremo le ipotesi di stazionarietà e di assenza di memoria che hanno consentito di apportare semplificazioni rilevanti al modello di funzionamento asintotico della sorgente. Si tenga però conto che tali ipotesi non sono sempre verosimili nell'ambito reale: se p.es. si devono codificare delle frasi in un linguaggio naturale non si può certo considerare il processo senza memoria: si pensi per esempio all'immediato successo nella ricostruzione di una parola come "q∗alsiasi", che si avvale evidentemente del fatto che una "q" (in italiano) è sempre seguita da una "u". Anche la stazionarietà non ha spesso riscontro nei casi pratici: pensando p.es. a una successione di frasi ricavate da

libri diversi (scritti in una qualunque lingua naturale), è ragionevole prevedere che nella transizione tra argomenti diversi si realizzi anche un nuovo assetto nella distribuzione di probabilità che modella la "generazione" del libro da parte della sorgente. Ciò nonostante, lo studio delle sorgenti sotto queste ipotesi restrittive è molto utile, poiché una volta stabilite le proprietà fondamentali le ipotesi possono essere rilassate progressivamente, fornendo così delle stime, più o meno accurate, del funzionamento della sorgente anche in condizioni diverse dalla stazionarietà e assenza di memoria. Non si trascuri inoltre la possibilità di individuare contesti particolari per i quali una o l'altra delle ipotesi possa valere; per quanto riguarda la mancanza di stazionarietà si può p.es. effettuare una "segmentazione" delle sequenze che escono dalla sorgente in blocchi per i quali l'ipotesi appaia invece verosimile.

Come preannunciato nel capitolo introduttivo, lo scopo della *codifica di sorgente* è quello di effettuare una traduzione dall'alfabeto della sorgente, che in generale può essere assimilato all'alfabeto di un linguaggio naturale, a quello di funzionamento del canale, che è legato a considerazione di carattere tecnico-operativo (per esempio alla circostanza che la tecnologia elettronica, che incarna il funzionamento dei varî dispositivi, è essenzialmente legata a una logica binaria, e quindi a un alfabeto di due simboli).

3.4.1 Funzione di codifica

Sia allora $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ l'alfabeto della sorgente, o alfabeto *primario*, e $\mathcal{B} = \{b_1, b_2, \ldots, b_D\}$ l'alfabeto sul quale si basa il funzionamento del canale, o alfabeto *secondario*. Di solito $D = 2$ e $K > D$. La traduzione tra le sequenze sui due alfabeti deve avvenire in modo tale che dalla sequenza secondaria che esce dal codificatore si riesca a ricavare in modo univoco la sequenza primaria generata dalla sorgente; tale condizione si esprime affermando che il codice dev'essere *univocamente decodificabile* (u.d.). È ovvio che l'ipotesi di univoca decodificabilità risulta irrinunciabile (ma si veda il caso dei codici *B-B* della sezione 3.5).

Esigenze di carattere economico richiedono inoltre che la traduzione sia accompagnata da una efficienza che si può misurare sulla base del rapporto tra lunghezza (media) della sequenza secondaria e lunghezza (media) della sequenza primaria (o viceversa). Questa efficienza non è di per sé irrinunciabile, ma sicuramente fortemente auspicabile. Infatti l'occupazione del canale di trasmissione ha un costo che si può assimilare proporzionale alla lunghezza della sequenza secondaria. Anche nel caso di *memorizzazione* delle informazioni il costo della stessa è proporzionale alla quantità di memoria accupata, cioè di nuovo alla lunghezza della sequenza secondaria. Il problema da risolvere nell'ambito della codifica di sorgente è allora quello di individuare una strategia di traduzione u.d. che comporti un'elevata efficienza, cioè un'elevata *compressione dei dati*.

Passiamo ora alla descrizione del modello vero e proprio. Sia $\mathcal{A}^+$ l'insieme di tutte le sequenze finite (stringhe) costruite con elementi di $\mathcal{A}$ e $*$ la legge di composizione interna per i suoi elementi data dalla *concatenazione* tra gli stessi; così se $\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2} \in \mathcal{A}^+$, anche $\boldsymbol{\alpha_1} * \boldsymbol{\alpha_2} \in \mathcal{A}^+$. Se consideriamo ora l'insieme $\mathcal{B}^+$ associato all'alfabeto $\mathcal{B}$ delle sequenze secondarie si definisce *codice astratto* una qualunque applicazione

$$\varphi : \mathcal{A}^+ \to \mathcal{B}^+$$

Questa definizione è però troppo generale e porta a un codice impraticabile, poiché pone in corrispondenza gli elementi di due insiemi infiniti. Tuttavia essa può essere circostanziata al caso in cui la φ sia un *omomorfismo*; se ciò accade si ha una semplificazione notevole, poiché in tal caso $\varphi(\boldsymbol{\alpha_1} * \boldsymbol{\alpha_2}) = \varphi(\boldsymbol{\alpha_1}) * \varphi(\boldsymbol{\alpha_2})$ (si assume che anche $\mathcal{B}^+$ abbia come legge di composizione interna la $*$). In altre parole la *codifica* di una sequenza di stringhe primarie concatenate si ottiene mediante *concatenazione delle codifiche* (basate sull'alfabeto secondario) delle singole stringhe primarie. Come conseguenza di ciò verificheremo che è sufficiente stabilire una corrispondenza tra gli elementi di due insiemi finiti, quello dei *messaggi*, $\mathcal{M} = \{m_1, m_2, \ldots, m_{_T}\}$, che contiene stringhe (eventualmente a lunghezza variabile) costruite sulle lettere di $\mathcal{A}$, e l'insieme dei *valori* (o *parole di codice*) che la φ fa assumere a tali messaggi. Ecco allora che accanto all'insieme primario $\mathcal{M} = \{m_i\}_{i=1}^{K}$ considereremo il *dizionario*

$$\mathcal{W} = \{\varphi(m_i)\}_{i=1}^{K}$$

delle *parole di codice* (eventualmente a lunghezza variabile) associate a $\mathcal{M}$. In particolare i messaggi possono ridursi a singole lettere, e in questo caso il dizionario rappresenta la codifica dell'alfabeto primario; la costruzione di $\mathcal{M}$ è allora immediata, poiché $\mathcal{M} = \mathcal{A}$.

Def. 3.7. *Si definisce codice la terna* $\mathcal{C} = \{\mathcal{M}, \varphi, \mathcal{W}\}$.

Si noti che la costruzione di $\mathcal{M}$ non è in generale banale: bisogna infatti fare in modo che con i suoi elementi si possa comporre una qualunque stringa $\boldsymbol{\alpha} \in \mathcal{A}^+$; ciò equivale a dire che qualunque sia la sequenza $\boldsymbol{\alpha}$ emessa dalla sorgente deve essere possibile effettuare una sua *segmentazione* negli elementi di $\mathcal{M}$.

Def. 3.8. *Una famiglia* $\mathcal{M} = \{m_1, m_2, \ldots, m_{_T}\}$ *di messaggi si dice esauriente se, qualunque sia la successione* $\boldsymbol{\alpha}$ *semi-infinita a destra costruita con gli elementi di* $\mathcal{A}$*, esiste sempre almeno un prefisso di* $\boldsymbol{\alpha}$ *che appartiene a* $\mathcal{M}$.

L'univoca decodificabilità può essere introdotta imponendo *l'iniettività* della funzione di codifica φ

Def. 3.9. *Un codice* $\mathcal{C} = \{\mathcal{M}, \varphi, \mathcal{W}\}$ *è univocamente decodificabile se,* $\forall\, \boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \mathcal{A}^+$, $\boldsymbol{\alpha} \neq \boldsymbol{\beta}$ *implica* $\varphi(\boldsymbol{\alpha}) \neq \varphi(\boldsymbol{\beta})$.

Per quanto attiene i codici di sorgente distingueremo gli stessi sulla base della lunghezza (costante o variabile) dei messaggi e delle parole di codice. Ci sono dunque quattro possibilità:

Codici blocco-blocco (*B-B*). In questo caso i messaggi si riducono a blocchi di lunghezza costante pari a n, cioè $\mathcal{M} = \mathcal{A}^n$ per $n \geq 1$; anche la lunghezza delle parole di codice è costante;

Codici blocco-lunghezza variabile (*B-LV*). Le parole di codice $\varphi(m_i)$ sono a lunghezza variabile, mentre i messaggi hanno lunghezza costante pari a n. Molto spesso $n = 1$, e la codifica a lunghezza variabile riguarda le singole lettere di $\mathcal{A}$. Se $n > 1$ allora i messaggi sono tutte le possibili K^n n-ple di $\mathcal{A}^n$.

Codici lunghezza variabile-blocco (*LV-B*). I messaggi sono a lunghezza variabile e bisogna garantire l'esaurienza. Le parole di codice sono invece a lunghezza costante.

Codici lunghezza variabile-lunghezza variabile (*LV-LV*). È il caso più generale: a ciascun messaggio primario di lunghezza variabile si fa corrispondere una parola di codice anch'essa di lunghezza variabile.

Esempio 3.2. Sia $\mathcal{A} = \{a, b, c\}$ e $\mathcal{B} = \{0, 1\}$. Effettuiamo una codifica per ciascuno dei quattro casi precedenti, supponendo che la sequenza emessa sia $\boldsymbol{\alpha} = bbacab$.

B-B Sia $\mathcal{W} = \{\varphi(a), \varphi(b), \varphi(c)\} = \{00, 01, 10\}$. La codifica va fatta sulle singole lettere: $\varphi(bbacab) = 01/01/00/10/00/01$.

B-LV Se $\mathcal{W} = \{\varphi(a), \varphi(b), \varphi(c)\} = \{0, 10, 11\}$, allora $\varphi(bbacab) = 10/10/0/11/0/10$.

LV-B In questo caso bisogna costruire una famiglia di messaggi a lunghezza variabile; poniamo p.es. $\mathcal{M} = \{aa, ab, ac, b, c\}$. La funzione di codifica associa ai messaggi parole di codice a lunghezza costante, p.es. $\mathcal{W} = \{\varphi(aa), \varphi(ab), \varphi(ac), \varphi(b), \varphi(c)\} = \{000, 001, 010, 011, 100\}$. La segmentazione della sequenza primaria è allora $b/b/ac/ab$ e la successione delle parole di codice emesse dal codificatore è $011/011/010/001$.

LV-LV Manteniamo la stessa famiglia di messaggi vista prima $\mathcal{M} = \{aa, ab, ac, b, c\}$. La funzione di codifica associa a ciascun messaggio parole di codice a lunghezza variabile, p.es. $\mathcal{W} = \{\varphi(aa), \varphi(ab), \varphi(ac), \varphi(b), \varphi(c)\} = \{000, 001, 01, 10, 11\}$. Con la stessa segmentazione in messaggi del caso precedente si ottiene la seguente successione secondaria: è $10/10/01/001$. ◯

Dall'esempio si noti come, nel caso in cui le parole di codice siano a lunghezza costante, l'omomorfismo della funzione di codifica sposti la richiesta di univoca decodificabilità dalle sequenze alle singole parole di codice; in tal caso è infatti sufficiente fare in modo che esse siano tutte distinte. Per i codici con parole a lunghezza variabile quest'ultima condizione è comunque necessaria, ma in generale non sufficiente: può infatti esistere un'ambiguità *strutturale* nell'esecuzione della decodifica, legata alla mancanza di iniettività della φ, che fa corrispondere a due (o più) sequenze primarie un'unica sequenza secondaria. Un esempio semplice è dato dal seguente codice *B-LV*: $\mathcal{W} = \{\varphi(a), \varphi(b), \varphi(c)\} = \{0, 01, 001\}$; la sequenza secondaria 000101 si può segmentare, e quindi decodificare, come $0/0/01/01 = aabb$, ma anche come $0/001/01 = acb$.

Oss. 3.1. Un discorso analogo vale anche per la *segmentazione* delle sequenze primarie nel caso di codici con messaggi a lunghezza variabile; la principale differenza operativa consiste nel fatto che in questo caso una segmentazione non univoca (mancanza di univoca *codificabilità*) non è pregiudizievole, poiché si può scegliere una delle due (o più) segmentazioni secondo opportuni criteri di convenienza (p.es. quello di avere una lunghezza massima per il messaggio; approfondiremo questo discorso nella sez. 3.4.8).

Oss. 3.2. Mentre la condizione u.d. è irrinunciabile per il dizionario di una codifica *B-LV*, la sua esaurienza non è rilevante, poiché la decodifica va fatta su sequenze costruite con elementi dello stesso dizionario. Viceversa l'esaurienza è strettamente necessaria per la famiglia di messaggi in una codifica *LV-B*, poiché dalla sorgente può uscire una sequenza qualunque. Per contro, come riferito nell'osservazione precedente, si può operare anche in assenza di univoca codificabilità.

3.4.2 Ritardi di (de)codifica

Un altro aspetto molto importante della struttura del codice è legato al *ritardo di decodifica* (o di codifica) che si deve sopportare ogniqualvolta non si riesce a riconoscere una parola di codice (un messaggio) subito dopo aver letto la sua ultima lettera. Se p.es. $\mathcal{W} = \{\varphi(a), \varphi(b), \varphi(c)\} = \{0, 01, 011\}$, l'unica segmentazione possibile per 001010 è $0/01/01/0$, che corrisponde a $abba$. Tuttavia quando riceviamo il primo 0, prima di procedere alla decodifica (o alla segmentazione) dobbiamo attendere di conoscere la seconda cifra; se questa è di nuovo 0 decodifichiamo con a, ma se riceviamo un 1 (e dunque la decodifica sarà b o c) dobbiamo attendere nuovamente la cifra successiva. In questo caso si dice che il codice è affetto da un *ritardo di (de)codifica* pari a 1. Si noti che esistono casi patologici di codici u.d. con ritardi di decodifica potenzialmente infiniti (si fa sempre riferimento al massimo ritardo possibile). Ne è esempio il codice $\mathcal{W} = \{\varphi(a), \varphi(b), \varphi(c)\} = \{1, 10, 00\}$, per il quale sequenze del tipo 10^n1, che

contengono n zeri tra due uni, vengono decodificate come $ac^{n/2}a$, oppure come $bc^{(n-1)/2}a$, a seconda che n sia pari o dispari. Anche se non esistono sequenze ambigue n può assumere a priori qualunque valore, e il ritardo di decodifica non è limitabile superiormente.

Come appare evidente dagli esempi precedenti si può evitare il ritardo di (de)codifica se (e solo se) si evitano di usare come parole di codice i *prefissi* di altre parole di codice.

Def. 3.10. *Una famiglia $\mathcal{M} = \{m_1, m_2, \ldots, m_T\}$ di parole di codice (messaggi) si dice a prefisso (o propria) se nessuna parola di codice è prefisso di ogni altra parola di codice (se nessun messaggio è prefisso di ogni altro messaggio).*

I codici che soddisfano questo requisito si definiscono *codici a prefisso*. Detto allora *istantaneo* un codice senza ritardo di (de)codifica vale la seguente

Proposizione 3.1. *Un codice è istantaneo se e solo se è a prefisso.*

Dim. Se il codice è a prefisso è evidente che è anche istantaneo. Per il viceversa si supponga che il codice sia istantaneo, ma che per assurdo esista $\boldsymbol{\alpha}$ prefisso di un'altra parola di codice $\boldsymbol{\beta} = \boldsymbol{\alpha\rho}$. Quando riceviamo una sequenza che contiene come prefisso $\boldsymbol{\alpha}$ non possiamo decidere sulla decodifica, poiché bisogna attendere di vedere se il suffisso è o meno $\boldsymbol{\rho}$. □

Oss. 3.3. Un codice a prefisso è anche *univocamente decodificabile*, poiché esiste un'unica segmentazione possibile, cioè quella che impone la decodifica della parola in corrispondenza del riconoscimento della sua ultima lettera. Naturalmente non tutti i codici u.d. sono a prefisso.

Anche sulla base di quest'ultima osservazione risulta chiara la centralità della proprietà del prefisso.

3.4.3 Alberi di codice

La descrizione del dizionario $\mathcal{W}$ del codice (o della famiglia $\mathcal{M}$ di messaggi) può essere fatta avvalendosi degli *alberi radicati*, che sono grafi connessi privi di cicli nei quali viene posto in evidenza un nodo detto *radice*. Se l'alfabeto di riferimento ha cardinalità D useremo un albero radicato D-ario, nel quale da ogni nodo escono al più D rami. Se da un nodo non escono rami esso si dice *nodo terminale* o *foglia*, altrimenti è detto *nodo non terminale* o *interno*.

È conveniente contrassegnare i rami uscenti da ogni nodo con le lettere dell'alfabeto D-ario, in modo che un qualunque cammino che dalla radice si estenda a un nodo qualunque possa essere rappresentato in modo univoco mediante la successione dei contrassegni. Così facendo si può associare a ciascuna parola di codice (o a ciascun messaggio) $b_{i_1} b_{i_2} \ldots b_{i_n}$ un nodo dell'albero secondo la seguente

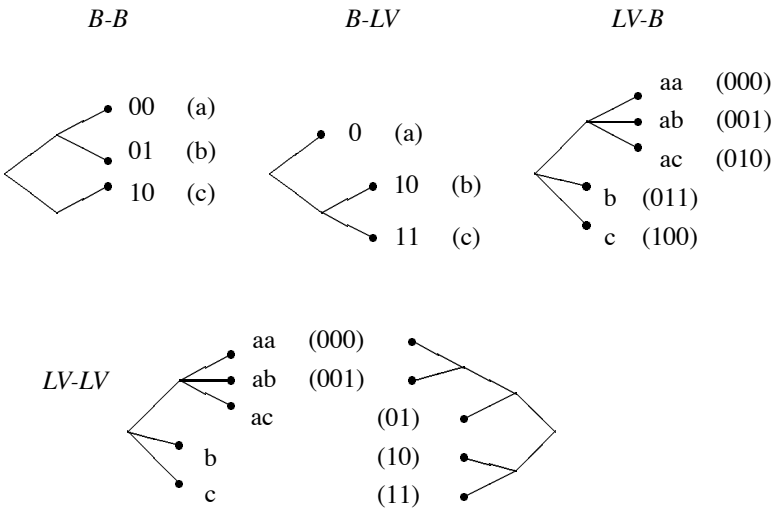


Figura 3.1 Alberi di codice relativi all'esempio 3.2.

Procedura di assegnazione dei nodi. Usando le lettere $b_1, b_2, \ldots, b_D$ di $\mathcal{B}$ come contrassegni per i rami, il nodo relativo a una parola di codice $\boldsymbol{\alpha} = b_{i_1} b_{i_2} \ldots b_{i_n}$ è quello terminale del cammino $b_{i_1} b_{i_2} \ldots b_{i_n}$ effettuato partendo dalla radice.

Ricordiamo inoltre che un nodo $\boldsymbol{\alpha}$ è *predecessore* di un nodo $\boldsymbol{\beta}$ se esso appartiene al cammino tra la radice e $\boldsymbol{\beta}$; in tal caso si dice che $\boldsymbol{\beta}$ è *successore* di $\boldsymbol{\alpha}$. Un nodo $\boldsymbol{\beta}$ che sia immediato successore di $\boldsymbol{\alpha}$ si dice *figlio*, mentre $\boldsymbol{\alpha}$ si dice suo *genitore*; due figli dello stesso genitore si chiamano *gemelli*. Ricordiamo inoltre che *l'ordine* di un nodo corrisponde alla lunghezza del cammino che lo separa dalla radice.

Def. 3.11. *Un albero D-ario dal quale escano esattamente D rami per ciascun nodo non terminale si dice completo . Un albero completo con D^n nodi terminali (o foglie) di ordine n si dice zeppo.*

Le foglie di un albero zeppo di ordine n rappresentano messaggi (o parole di codice) a lunghezza costante. Se sono dei messaggi essi si possono interpretare come l'alfabeto $\mathcal{A}^n$ dell'estensione n-esima della sorgente. In figura 3.1 possiamo vedere gli alberi associati ai quattro codici dell'esempio 3.2. Nel seguito del paragrafo considereremo le proprietà *strutturali* degli insiemi $\mathcal{W}$ e $\mathcal{M}$. Queste ultime non dipendono dal fatto che essi vengano impiegati come famiglia di messaggi in una codifica *LV-B* o come dizionario di parole di codice nella codifica *B-LV*, ma sono legate essenzialmente alla *struttura* dell'albero di codice

associato all'insieme. L'albero descriverà allora una *famiglia dei messaggi* $\mathcal{M}$ nel caso di una codifica *LV-B* o *l'insieme delle parole di codice* del dizionario $\mathcal{W}$ nel caso di una codifica *B-LV*. Le considerazioni che seguono, e che riguardano l'univoca decodificabilità, l'esaurienza e i ritardi di decodifica, sono relative tanto ai messaggi che alle parole di codice; useremo l'una o l'altra terminologia a seconda che si abbia in mente come obiettivo una segmentazione primaria per un codice *LV-B* oppure una decodifica secondaria per un codice *B-LV*.

3.4.4 Univoca (de)codificabilità

La descrizione del codice mediante un albero consente di evidenziare facilmente tanto la proprietà del prefisso quanto quella dell'esaurienza. Se $\mathcal{W}$ è un dizionario e $l^* = \max_i\ l_i$ la massima lunghezza associata a una sua parola, si costruisca l'albero zeppo D-ario di ordine l^*. Si associ ora a ciascuna parola un nodo dell'albero, secondo la *Procedura di assegnazione dei nodi* vista precedentemente, conservando tutti i cammini che collegano la radice con i nodi così individuati. Eliminando tutti i rami e i nodi che non vengono toccati nei vari cammini si ottiene la struttura *dell'albero di codice* associata al dizionario $\mathcal{W}$. Per garantire l'esaurienza è necessario che per ciascuno dei percorsi che escono dalla radice si incontri sempre *almeno* una parola di $\mathcal{W}$. Al contrario, per la validità della proprietà del prefisso è necessario incontrare sempre *al più* una parola di $\mathcal{W}$. Se le due condizioni valgono contemporaneamente, per ogni percorso che si diparte dalla radice incontreremo una e una solo parola di codice; in tal caso la famiglia si dice *completa* .

Intuitivamente, dette $l_1, l_2, \ldots, l_K$ le lunghezze delle parole di codice, per garantire la condizione u.d. dobbiamo avere dei valori "non troppo piccoli" di l_i, in modo tale che sull'albero di codice le parole siano ben distanziate e isolate l'una dall'altra. Viceversa, per garantire l'esaurienza le lunghezze $m_1, m_2, \ldots, m_T$ dei messaggi devono essere "non troppo grandi" per evitare di lasciare dei varchi nei percorsi che partono dalla radice (la presenza di un varco consente la fuga all'infinito di un percorso e quindi determina l'impossibilità di trovare un prefisso in $\mathcal{M}$). In quest'ultimo caso i messaggi devono stare "addossati" alla radice, mentre per la condizione u.d. devono stare sufficientemente lontani.

Entrambe le situazioni hanno una precisa descrizione in termini matematici. Cominciamo con la celebre disuguaglianza di McMillan-Kraft, proposta originariamente da Kraft per i codici a prefisso ed estesa successivamente da McMillan per tutti i codici u.d.; facciamo riferimento alla dimostrazione proposta in un secondo tempo da Karush e adattata da Longo.

Teorema 3.9 (Disuguaglianza di McMillan-Kraft). *Se $l_1, l_2, \ldots, l_K$ sono le lunghezze delle parole di un codice u.d. allora*

$$\sum_{i=1}^{K} D^{-l_i} \leq 1 \tag{3.54}$$

Dim. Indichiamo con $N(n,h)$ il numero delle sequenze primarie di lunghezza n cui corrisponde una sequenza secondaria di lunghezza h. È evidente che $n \leq h \leq nl^*$, poiché $1 \leq l_i \leq l^*$ e dunque $N(n,h) = 0$ quando $h < n$ oppure $h > nl^*$. Inoltre

$$N(n+1,h) = \sum_{i=1}^{K} N(n, h-l_i) \tag{3.55}$$

poiché la sequenza di parole corrispondenti a $n+1$ lettere si ricava giustapponendo la parola per l'ultima lettera, che ha lunghezza l_i per qualche i, alla sequenza delle parole per le prime n lettere. La chiave della dimostrazione consiste nel verificare per induzione la seguente uguaglianza

$$\sum_{h=n}^{nl^*} N(n,h) D^{-h} = \left(\sum_{i=1}^{K} D^{-l_i} \right)^n$$

Per $n = 1$ la relazione è vera

$$\sum_{h=1}^{l^*} N(1,h) D^{-h} = \sum_{i=1}^{K} D^{-l_i} \qquad \text{poiché } N(1,h) = \begin{cases} 1 & \text{se } h = l_i \\ 0 & \text{altrimenti} \end{cases}$$

Supponiamola vera per n e dimostriamola per $n+1$:

$$\left(\sum_{i=1}^{K} D^{-l_i} \right)^{n+1} = \left(\sum_{h=n}^{nl^*} N(n,h) D^{-h} \right) \left(\sum_{i=1}^{K} D^{-l_i} \right) =$$

$$= \sum_{i=1}^{K} \sum_{h=n}^{nl^*} N(n,h) D^{-(h+l_i)} \overset{(1)}{=} \sum_{i=1}^{K} \sum_{t_i=n+l_i}^{nl^*+l_i} N(n, t_i - l_i) D^{-t_i} =$$

$$\overset{(2)}{=} \sum_{t=n+1}^{nl^*+l^*} D^{-t} \sum_{i=1}^{K} N(n, t-l_i) \overset{(3)}{=} \sum_{t=n+1}^{(n+1)l^*} N(n+1,t) D^{-t}$$

dove la $^{(1)}$ segue dal cambio di variabile $t_i = h + l_i$; per la $^{(2)}$ si ha $N(n, t_i - l_i) = 0$ quando $t_i < n + l_i$ oppure $t_i > nl^* + l_i$, il che consente di estendere la sommatoria aggiungendo termini nulli e di usare un'unica variabile t in luogo delle t_i. La $^{(3)}$ segue infine dalla (3.55). L'induzione risulta quindi dimostrata. D'altra parte, a seguito dell'ipotesi di u.d. si deve avere $N(n,h) \leq D^h$, cioè $N(n,h) D^{-h} \leq 1$, che porta a

$$\sum_{h=n}^{nl^*} N(n,h) D^{-h} \leq nl^* - (n-1) \leq nl^* \quad \text{cioè} \quad \left(\sum_{i=1}^{K} D^{-l_i} \right)^n \leq nl^*$$

Se ora fosse $\sum_{i=1}^{K} D^{-l_i} > 1$ per n sufficientemente grande l'ultima disuguaglianza non potrebbe valere. □

La condizione del teorema 3.9 oltre che necessaria è anche sufficiente per la costruzione di un codice u.d. (a prefisso), così come specificato dal seguente

Teorema 3.10. *Se l'insieme dei numeri $D, l_1, l_2, \ldots, l_K$ soddisfa la disuguaglianza di McMillan-Kraft, allora esiste un codice u.d. (a prefisso) le cui parole w_i hanno lunghezza l_i.*

Dim. Si supponga che sia $l_1 \leq l_2 \leq \ldots \leq l_K$ e si scelga un nodo ν_1, di ordine l_1, sull'albero D-ario zeppo di ordine l_K. Indichiamo con w_1 la parola costruita con il contrassegno di ν_1. Se elidiamo il sottoalbero generato da w_1, e contenente $D^{l_K - l_1}$ nodi terminali (foglie), dalla disuguaglianza di McMillan-Kraft, moltiplicando per D^{l_K}, si ottiene $\sum_{i=1}^{K} D^{l_K - l_i} \leq D^{l_K}$, cioè $D^{l_K - l_1} < D^{l_K}$; ci sono dunque foglie che non hanno w_1 come prefisso. Nell'albero potato scegliamo ora ν_2, di ordine l_2, e indichiamo con w_2 la parola basata sul contrassegno di ν_2. Se elidiamo il sottoalbero generato da w_2, e contenente $D^{l_K - l_2}$ nodi terminali, sempre per la disuguaglianza di McMillan-Kraft si ha $D^{l_K - l_1} + D^{l_K - l_2} < D^{l_K}$. Il procedimento può essere ripetuto finché si arriva alla scelta dell'ultima parola di codice, che potrà essere effettuata giacché $D^{l_K - l_1} + D^{l_K - l_2} + \ldots + D^{l_K - l_{K-1}} < D^{l_K}$. □

La validità dei teoremi 3.9 e 3.10 porta come conseguenza immediata il seguente

Corollario 3.1. *Condizione necessaria e sufficiente per l'esistenza di un codice D-ario u.d. (a prefisso) è che valga la disuguaglianza di McMillan-Kraft.*

Oss. 3.4. La necessità può essere risolta più semplicemente quando si usi l'ipotesi di codifica a prefisso. Detta infatti l^* la lunghezza massimale, qualunque parola w_i non è prefisso di alcun altra parola di codice, e ciò accade anche per tutte le $D^{l^* - l_i}$ foglie generate da w_i sull'albero zeppo di ordine l^*. Di conseguenza, poiché le foglie sono D^{l^*}, deve accadere che $\sum_{i=1}^{K} D^{l^* - l_i} \leq D^{l^*}$, cioè la (3.54).

Rimane ora da affrontare il problema della verifica di u.d. per un codice preassegnato. A tal scopo si può impiegare il seguente procedimento, dovuto a Sardinas e Patterson:

Metodo di Sardinas-Patterson. Assegnato il dizionario $\mathcal{W} = \{w_1, w_2, \ldots, w_K\}$ (la famiglia di messaggi $\mathcal{M} = \{m_1, m_2, \ldots, m_T\}$) del codice si costruisca una successione $R_0, R_1, R_2, \ldots$ di insiemi nel modo seguente: $R_0 = \mathcal{W}$ $(= \mathcal{M})$. Per costruire R_1 si confrontano tutte le coppie di R_0; se esiste w_i tale che $w_j = w_i \rho_{ij}^{(1)}$ si pone il suffisso $\rho_{ij}^{(1)}$ in R_1, che risulta essere dunque l'insieme di tutti i suffissi ottenuti confrontando coppie di parole (messaggi) di R_0. In generale si costruisce R_n verificando se

se qualche elemento w_i di R_0 è prefisso di qualche elemento di R_{n-1} e viceversa:

$$\left.\begin{array}{ll} \rho_k^{(n-1)} = w_i \rho_{ik}^{(n)} & w_i \in R_0 \\ w_i = \rho_k^{(n-1)} \rho_{ik}^{(n)} & \rho_k^{(n-1)} \in R_{(n-1)} \end{array}\right\} \Longrightarrow \text{si pone } \rho_{ik}^{(n)} \text{ in } R_n$$

Si osservi che l'insieme degli R_n potrebbe essere infinito, come vedremo in un esempio successivo.

Teorema 3.11. *Condizione necessaria e sufficiente affinchè un codice sia univocamente decodificabile (una famiglia di messaggi sia univocamente codificabile) è che nessuno degli insiemi R_n con $n \geq 1$ contenga parole di codice (messaggi) di R_0.*

Si osservi che se l'ultimo insieme che si costruisce è vuoto, allora il codice è anche a prefisso. Per la dimostrazione del teorema rimandiamo il lettore a [3].

Esempio 3.3. Consideriamo i seguenti tre esempi: $\mathcal{W}_1 = \{10, 010, 1, 1110\}$, $\mathcal{W}_2 = \{0, 001, 101, 11\}$,$\mathcal{W}_3 = \{0, 2, 03, 011, 104, 341, 11234\}$ Si noti che $\mathcal{W}_1$

R_0	R_1	R_2
10	0	10
010	110	
1		
1110		

$\mathcal{W}_1 \longrightarrow$ non u.d.

R_0	R_1	R_2	R_3	R_4	...
0	01	1	1	1	...
001			01	01	...
101					
11					

$\mathcal{W}_2 \longrightarrow$ u.d.

R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7
0	3	41	34	1	04	4	$\emptyset$
2	11	234			1234		
03							
011							
104							
341							
11234							

$\mathcal{W}_3 \longrightarrow$ u.d.

non è u.d., in quanto 10 sta nel dizionario. $\mathcal{W}_2$ invece è u.d., anche se c'è un numero infinito di insiemi R_n. Per quanto riguarda infine $\mathcal{W}_3$ si può notare che R_7 è vuoto: ciò garantisce che si tratta di codice a prefisso. ◯

3.4.5 Esaurienza e completezza

La proprietà dell'esaurienza richiede che per qualunque cammino uscente dalla radice si incontri sempre almeno un messaggio. Ciò consente di effettuare una segmentazione a lunghezza variabile di una qualunque successione costruita con elementi dell'alfabeto $\mathcal{A}$. Questa condizione risulta dunque irrinunciabile se si vuole effettuare una codifica da lunghezza-variabile a blocco. Si noti che se vale anche la proprietà del prefisso la segmentazione è unica e si evitano i ritardi di codifica, così come specificato dalla proposizione 3.1.

Esempio 3.4. La famiglia $\mathcal{M} = \{aaa, aa, a, b, ba, c\}$ è esauriente, poiché ogni successione costruita sull'alfabeto ternario $\mathcal{A} = \{a, b, c\}$ ha come prefisso a, b o c (che sono elementi di $\mathcal{M}$). Tuttavia essa non è a prefisso, poiché a è prefisso di aa e di aaa; inoltre b è prefisso di ba. Ciò implica un ritardo e una ambiguità nella segmentazione. La sequenza $aaabac$ si può infatti segmentare come $a-a-a-b-a-c$, $aaa-ba-c$ o anche come $a-aa-ba-c$ ecc. In altre parole la famiglia non è univocamente (de)codificabile.

Ribadiamo che la mancanza della proprietà del prefisso per una famiglia esauriente associata a una codifica *LV-B* non è di per sé pregiudizievole, in quanto si può decidere di segmentare quando si raggiunge il primo prefisso che sta nella famiglia, oppure di usare il prefisso più lungo disponibile; tuttavia essa è auspicabile, in quanto annulla i ritardi di codifica.

L'esaurienza impone la validità di una disuguaglianza che appare come la controparte della disuguaglianza di McMillan-Kraft per l'univoca decodificabilità.

Teorema 3.12. *Siano $n_1, n_2, \ldots, n_T$ le lunghezze dei messaggi $\mathcal{M} = \{m_1, m_2, \ldots, m_T\}$ di una famiglia esauriente costruiti con le lettere di un alfabeto K-ario $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$. Allora vale la seguente disuguaglianza*

$$\sum_{i=1}^{T} K^{-n_i} \geq 1 \tag{3.56}$$

Dim. Si prenda $n \geq n_i \ \forall i$ e si consideri l'albero zeppo K-ario di ordine n. Le K^n foglie sono prefisso di una qualunque successione (di lunghezza $\geq n$). Il messaggio m_i di lunghezza n_i è allora prefisso di K^{n-n_i} inizi di sequenza. Per l'esaurienza deve verificarsi che

$$\sum_{i=1}^{T} K^{n-n_i} \geq K^n$$

cioè la tesi. □

La contemporanea validità dell'esaurienza e della proprietà del prefisso comporta il seguente

Corollario 3.2. *Se $n_1, n_2, \ldots, n_T$ sono le lunghezze dei messaggi di una famiglia completa su un alfabeto di K simboli allora*

$$\sum_{i=1}^{T} K^{-n_i} = 1 \tag{3.57}$$

Dim. Immediata dalla contemporanea validità delle (3.54) e (3.56) □

Esempio 3.5.

- Sia $\mathcal{A} = \{a, b, c\}$. La famiglia $\mathcal{M}_e = \{a, aa, ab, ac, b, c\}$ è esauriente, ma non a prefisso. Per essa $\sum_{i=1}^{6} 3^{-n_i} \geq 1$. Si veda la figura 3.2(1), dove il simbolo ◉ individua un nodo che viola la proprietà del prefisso.
- $\mathcal{M}_p = \{aa, ab, ac, c\}$ è a prefisso, ma non esauriente (fig.3.2(2)). Dunque $\sum_{i=1}^{4} 3^{-n_i} \leq 1$. Il ramo tratteggiato indica una via di fuga.
- La famiglia $\mathcal{M}_{ep} = \{aa, ab, ac, b, c\}$ è invece esauriente e a prefisso, cioè completa (fig.3.2(3)). Dunque $\sum_{i=1}^{5} 3^{-n_i} = 1$ e le parole del suo dizionario vengono collocate sui nodi terminali di un albero completo.
- I messaggi della $\mathcal{M}_{nep} = \{aa, ac, aca, acc, c\}$ non godono invece di alcuna proprietà. Non si può dunque dir nulla sulla sommatoria (fig.3.2(4)). ◯

Una famiglia completa si associa in modo ovvio a un albero completo, e la sua costruzione prende spunto proprio dalla struttura dell'albero.

Def. 3.12. *Sia $\mathcal{M}(T)$ una famiglia completa di T messaggi e sia $m^* \in \mathcal{M}(T)$. La famiglia*

$$\mathcal{M}(T + K - 1) = \left\{ (\mathcal{M}(T) - m^*) \bigcup_{i=1}^{K} m^* a_i \right\} \tag{3.58}$$

che si ottiene eliminando m^* e sostituendolo con le concatenazioni del tipo $m^*\ a_i$ $(1 \leq i \leq K)$, si dice estensione di $\mathcal{M}(T)$ mediante m^*.

La figura 3.3 mostra un esempio di un'estensione da $\mathcal{M}(5) = \{aa, ab, ac, b, c\}$ a $\mathcal{M}(5 + 3 - 1) = \{aa, ab, ac, b, ca, cb, cc\}$, avendo scelto il messaggio c per l'estensione. Per le famiglie complete $\mathcal{M}_j$ ottenute mediante j estensioni basate sulla (3.58) valgono le seguenti proprietà

Proposizione 3.2.

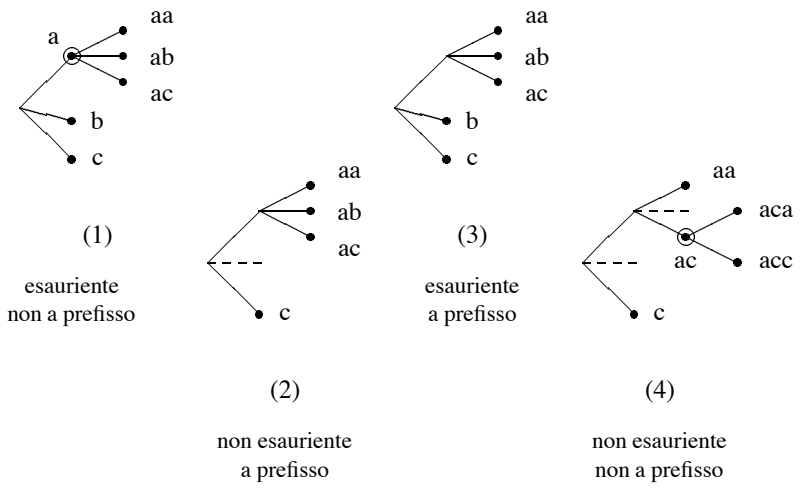


Figura 3.2 Alberi di codice relativi all'esempio 3.5.

1. *L'unica famiglia completa di K messaggi su $\mathcal{A}$ è $\mathcal{M}_0 = \mathcal{A}$.*

2. *Se $\mathcal{M}_j$ è completa, una sua estensione mediante un suo qualunque messaggio m^* è ancora completa.*

3. *Qualunque famiglia completa si ottiene mediante estensioni consecutive a partire da $\mathcal{A}$.*

4. *La cardinalità di una qualunque famiglia completa $\mathcal{M}_j$ è nella forma $T(j) = K + j(K-1)$, per qualche valore j del numero di estensioni.*

Dim. La *1.* è ovvia. Anche la *2.* è evidente, poiché all'atto dell'estensione viene mantenuta tanto l'esaurienza che la proprietà del prefisso; tutti i cammini che prima passavano per m^* ora passano per una delle concatenazioni m^*a_i, mentre se m^* non era prefisso di alcun messaggio nella vecchia famiglia a maggior ragione non lo saranno gli m^*a_i. Per la *3.* si consideri invece un messaggio $m_M = a_{j_1}a_{j_2}\ldots a_{j_M}$ di lunghezza massimale appartenente a $\mathcal{M}_j$. Poiché la famiglia è esauriente e a prefisso, essa deve avere come componenti anche tutti i gemelli di m_M, cioè messaggi che differiscono solamente per l'ultima lettera a_{j_M}, e nessun prefisso di questi. Se riduciamo la famiglia, togliendo tutti i K gemelli e sostituendoli col nodo genitore, la famiglia contratta $\mathcal{M}_{j-1}$ risulta ancora completa, e per essa si può ripetere il ragionamento ed effettuare una nuova

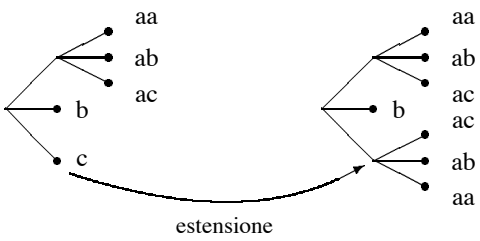


Figura 3.3 Estensione della famiglia $\{aa, ab, ac, b, c\}$ mediante il nodo c.

riduzione. Il procedimento può essere iterato fin quando si arriva alla più piccola famiglia completa, cioè $\mathcal{M}_0 = \mathcal{A}$; ricostruendo l'albero con la successione inversa delle contrazioni si giunge alla tesi. La *4.* si verifica immediatamente con l'induzione. □

3.4.6 Tasso di codifica

Come specificato all'inizio del capitolo, lo scopo della *codifica di sorgente* è quello di effettuare una traduzione dall'alfabeto primario della sorgente $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ all'alfabeto secondario $\mathcal{B} = \{b_1, b_2, \ldots, b_D\}$ sul quale si basa il funzionamento del canale. Per ciascun messaggio (o singola lettera) costruito con elementi di $\mathcal{A}$ e uscente dalla sorgente associeremo una parola di codice basata su elementi di $\mathcal{B}$. Le condizioni che regolano tale traduzione sono state specificate nelle sezioni precedenti; ora dobbiamo preoccuparci della sua efficienza. Tale valutazione si può effettuare in termini semplici se si fa riferimento alla categoria delle sorgenti *ergodiche* della definizione 3.5, per le quali le medie temporali coincidono, nel senso delle leggi dei grandi numeri, con le medie d'insieme. In tal caso infatti il "costo" di una sequenza, nelle ipotesi di stazionarietà e assenza di memoria, si può ritenere coincidente con la speranza matematica (o valor medio) della lunghezza dei messaggi (o delle parole di codice), almeno fin quando si presuppone che tutte le parole (o messaggi) abbiano un costo proporzionale alla loro lunghezza (ipotesi che può essere rimossa per formulare un modello più generale). Considereremo dunque

$$E[N] = \sum_{m \in \mathcal{M}} q(m)n(m) \qquad E[L] = \sum_{w \in \mathcal{W}} p(w)l(w) \tag{3.59}$$

dove $\mathcal{M}$ e $\mathcal{W}$ sono l'insieme dei messaggi e il dizionario, mentre $q(m), n(m)$ e $p(w), l(w)$ rappresentano rispettivamente probabilità e lunghezza dei messaggi e

delle parole di codice. La valutazione dell'efficienza nella traduzione è associata alla minimazione del *tasso*

$$R = \frac{E[L]}{E[N]} = \frac{\sum_{w \in \mathcal{W}} p(w) l(w)}{\sum_{m \in \mathcal{M}} q(m) n(m)} \tag{3.60}$$

o alla massimazione del *rapporto di compressione*

$$RC = \frac{1}{R} = \frac{E[N]}{E[L]} = \frac{\sum_{m \in \mathcal{M}} q(m) n(m)}{\sum_{w \in \mathcal{W}} p(w) l(w)} \tag{3.61}$$

Un codice per il quale il tasso (3.60) sia minimo (il rapporto di compressione sia massimo) si dice *ottimo*. Inoltre, accanto al *rapporto tra valori medi* (delle lunghezze) si potrebbe anche considerare *il valor medio del rapporto*

$$R' = E\left[\frac{L}{N}\right] \qquad RC' = E\left[\frac{N}{L}\right] \tag{3.62}$$

anche se in letteratura si è affermata l'impostazione (3.60).

Oss. 3.5. La ricerca dell'ottimalità per i due casi (3.60) e (3.62) potrebbe portare, in linea di principio, a codici diversi.

Assegnata allora una sorgente ergodica, in particolare stazionaria e senza memoria, il problema da risolvere per effettuare una buona *codifica di sorgente* è il seguente

Codifica di sorgente. Individuare una famiglia $\mathcal{M}$ di messaggi e un dizionario $\mathcal{W}$ in modo che il codice $\mathcal{C} = \{\mathcal{M}, \varphi, \mathcal{W}\}$ sia *univocamente decodificabile e ottimo*.

È evidente che, a seconda della tipologia di codice (*B-LV*, *LV-B*, *B-B*, *LV-LV*) avremo diverse soluzioni.

Prima di affrontare il problema dell'individuazione dei codici ottimi nei vari casi, studieremo alcune limitazioni normative che fissano il campo di esistenza per i codici u.d. Tali risultati derivano dai lavori di Shannon, o sono immediatamente riconducibili ad essi.

3.4.7 Codifica *B-LV*

Nella codifica da blocco a lunghezza variabile la lunghezza dei messaggi è costante, pari a n, mentre le parole di codice hanno lunghezza variabile. I messaggi $\boldsymbol{m}$ sono costituiti da tutte le possibili K^n n-ple di $\mathcal{A}^n = \mathcal{A} \times \mathcal{A} \times \ldots \times \mathcal{A}$, costruite sull'alfabeto $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$. In questo caso si fa dunque riferimento all'n-ima *estensione* $\mathcal{S}^n$ della sorgente $\mathcal{S}$. Nelle ipotesi di stazionarietà e

di assenza di memoria, se $P = \{p_1, p_2, \dots, p_K\}$ è la d.p. associata al funzionamento della sorgente la probabilità di un messaggio $\boldsymbol{m} = a_{i_1} a_{i_2}, \dots a_{i_n}$ è data da $q(m) = p_{i_1} p_{i_2}, \dots p_{i_n}$. Un caso particolare si ha quando $n = 1$, quando cioè si effettua una codifica delle singole lettere della $\mathcal{S}$. Consideriamo questo caso. Sia assegnato un codice per le singole lettere della sorgente, e sia $\mathcal{W} = \{w_1, w_2, \dots, w_K\}$ il dizionario del codice $\mathcal{C}$; le parole di codice hanno lunghezza $\mathcal{L} = \{l_1, l_2, \dots, l_K\}$, dove $l_j = |w_j|$. Il tasso della (3.60) corrisponde nel caso $n = 1$ alla *lunghezza media* delle parole di codice

$$E[L] = \sum_{i=1}^{K} p_i l_i \tag{3.63}$$

Vale allora il seguente fondamentale

Teorema 3.13 (Teorema di Shannon). *La lunghezza media delle parole di un arbitrario codice D-ario u.d. per le singole lettere di una sorgente $\mathcal{S}$ stazionaria e senza memoria è inferiormente limitata dall'entropia (D-aria) della sorgente*

$$E[L] \geq H_D(P) = -\sum_{i=1}^{K} p_i \log_D p_i \tag{3.64}$$

Dim. Si consideri la d.p. Q, dove $q_i = \frac{D^{-l_i}}{\sum_{j=1}^{K} D^{-l_j}} = \frac{D^{-l_i}}{\alpha}$, con $\alpha = \sum_{j=1}^{K} D^{-l_j}$. Calcoliamo ora la divergenza informazionale (2.3) $D(P//Q)$ tra P e Q, che è un quantità sempre non negativa, usando il logaritmo in base D.

$$\sum_{i=1}^{K} p_i \log_D \left[\frac{p_i}{D^{-l_i}}\alpha\right] = \sum_{i=1}^{K} p_i \log_D p_i + \left(\log_D D\right) \sum_{i=1}^{K} p_i l_i + \log_D \alpha \geq 0$$

$$\sum_{i=1}^{K} p_i l_i \geq -\sum_{i=1}^{K} p_i \log_D p_i - \log_D \alpha$$

e la tesi segue ora dal fatto che, a norma della disuguaglianza (3.54) di McMillan-Kraft, $\alpha \leq 1$. □

Oss. 3.6. Il valore minimo per $E[L]$ lo si ottiene quando $p_i = D^{-l_i}$, il che porta all'uguaglianza tra lunghezza media ed entropia. Si sarebbe pervenuti allo stesso risultato impostando una minimazione della lunghezza media che tenga conto del vincolo dato dalla disuguaglianza di McMillan-Kraft e basata sui moltiplicatori di Lagrange (si veda p.es. [19]). In ogni caso l'ipotesi $p_i = D^{-l_i}$ non si realizza quasi mai nella pratica, in quanto la P è assegnata a priori e dipende dalle caratteristiche statistiche della sorgente; è dunque alquanto improbabile che le p_i siano esprimibili come esponenti negativi di D (in tal caso si parla di d.p. *diadica*).

Oss. 3.7. Se la d.p. della sorgente è uniforme si ha $H_D(P) = \log_D K$, e non è possibile effettuare alcuna compressione. Infatti in tal caso si ottiene $E[L] \geq \log_D K$; ma $\log_D K$ è esattamente la lunghezza in D-it delle lettere primarie.

La quantità $r = E[L] - H_{_D}(P)$ si chiama *ridondanza* del codice, e a norma dell'osservazione precedente è di solito strettamente maggiore di zero. La ridondanza si può esprimere anche come una divergenza informazionale tra P e $\{D^{-l_i}\}_{i=1}^{K}$ quando quest'ultima diventi una distribuzione di probabilità

$$r(\mathcal{C}) = E[L(\mathcal{C})] - H_{_D}(P) = \sum_{i=1}^{K} p_i l_i + \sum_{i=1}^{K} p_i \log_{_D} p_i = \sum_{i=1}^{K} p_i \log_{_D} \frac{p_i}{D^{-l_i}} \tag{3.65}$$

Cioè accade se vale la (3.57) e il codice, oltre ad essere u.d., è anche esauriente. La ridondanza verrà ripresa più avanti, nell'interpretazione geometrica della codifica *B-LV* (si veda la sez. 5.5.1).

Dal contesto del teorema di Shannon non è chiaro se la limitazione specificata nella (3.64) sia stretta o possa essere migliorata; per verificarlo bisogna vedere se esistono dei codici con un tasso sufficientemente prossimo all'entropia. Un metodo semplice per costruirne uno è quello di imporre $E[L] \simeq H_{_D}(P)$, facendo in modo che sia $l_i \simeq -\log_{_D} p_i$. Poiché le lunghezze sono numeri interi mentre $-\log_{_D} p_i$ è in genere un numero reale, dobbiamo prendere la parte intera superiore $l_i = \lceil -\log_{_D} p_i \rceil$, dovendo comunque garantire la (3.64). Vale allora il seguente

Teorema 3.14. *Assegnata una sorgente stazionaria e senza memoria $\mathcal{S}$, esiste un codice D-ario a prefisso per le singole lettere di $\mathcal{S}$ la cui lunghezza media soddisfa la seguente disuguaglianza*

$$E[L] < H_{_D}(P) + 1 \tag{3.66}$$

Dim. A ogni lettera a_i di probabilità p_i si assegni una parola di codice di lunghezza $l_i = \lceil -\log_{_D} p_i \rceil \geq -\log_{_D} p_i$. Le lunghezze soddisfano la disuguaglianza di McMillan-Kraft, in quanto $D^{-l_i} \leq p_i$ e sommando su i si ottiene $\sum_{i=1}^{K} D^{-l_i} \leq \sum_{i=1}^{K} p_i = 1$. Dunque, a norma del teorema (3.10) si può costruire un codice a prefisso le cui parole hanno lunghezze l_i. Ma $l_i = \lceil -\log_{_D} p_i \rceil < -\log_{_D} p_i + 1$. Moltiplicando per p_i e sommando su i si ottiene $\sum_{i=1}^{K} p_i l_i < -\sum_{i=1}^{K} p_i \log_{_D} p_i + \sum_{i=1}^{K} p_i = H_{_D}(P) + 1$. □

Oss. 3.8. La limitazione (3.66) è la migliore possibile, poiché ci sono casi in cui $E[L] \to H_{_D}(P) + 1$. Per provarlo basta codificare una sorgente binaria $P = (\epsilon, 1-\epsilon)$ quando $\epsilon \to 0$. Poiché la lunghezza media è comunque 1, ma $H(\epsilon) \to 0$ (la d.p. tende a diventare degenere) e si ottiene $[H_{_D}(P) + 1] - E[L] \to 0$.

Il procedimento costruttivo introdotto nel teorema porta al cosiddetto *codice di Shannon-Fano*, dal quale si comprende che per rendere minima la lunghezza media si dovranno assegnare *parole di codice corte alle lettere più probabili*.

Oss. 3.9. Quando si realizza una buona codifica di sorgente, utilizzando un codice che soddisfa il teorema 3.14, ci si avvicina al limite teorico dell'entropia. La sequenza in uscita al codificatore non è dunque ulteriormente comprimibile. Ciò implica che la d.p. per le lettere secondarie dev'essere praticamente uniforme.

Prendiamo ora in considerazione una sorgente estesa $\mathcal{S}^n$, associata all'alfabeto $\mathcal{A}^n$ e alla d.p. P^n. Possiamo considerare le n-ple di $\mathcal{A}^n$ come lettere di $\mathcal{S}^n$. Nel caso di stazionarietà e di assenza di memoria l'entropia $H_{_D}(\mathcal{S}^n)$ del vettore di n v.a. indipendenti e identicamente distribuite è regolata dalla (2.23)

$$H_{_D}(\mathcal{S}^n) = H_{_D}(X_1, X_2, \dots, X_n) = \sum_{i=1}^{n} H_{_D}(X_i/X_1, X_2, \dots, X_{i-1}) =$$
$$= \sum_{i=1}^{n} H_{_D}(X_i) = nH_{_D}(\mathcal{S}) \tag{3.67}$$

dove le ultime due uguaglianze derivano rispettivamente dall'ipotesi di assenza di memoria e stazionarietà. Applicando i teoremi precedenti alla $\mathcal{S}^n$ si ottiene

$$H_{_D}(\mathcal{S}^n) \leq E[L^n] < H_{_D}(\mathcal{S}^n) + 1$$

e tenuto conto della (3.67) si ricava

$$H_{_D}(\mathcal{S}) \leq \frac{E[L^n]}{n} < H_{_D}(\mathcal{S}) + \frac{1}{n} \tag{3.68}$$

che porta al celebre

Teorema 3.15 (Teorema di Shannon per sorgenti estese). *Sia $\mathcal{S}$ una sorgente stazionaria e senza memoria. Esiste allora una successione di codici D-ari per le estensioni n-esime di $\mathcal{S}$ il cui tasso $R_n = E[L^n]/n$ tende asintoticamente all'entropia $H_{_D}(\mathcal{S})$ della sorgente.*

Dim. Immediata, tenuto conto che $1/n \to 0$ quando $n \to \infty$ □

Il teorema 3.15 fa emergere il ruolo dell'entropia nell'ambito della codifica di sorgente, poiché lega una quantità operativa, come il tasso del codice, a una grandezza strutturale della sorgente, l'entropia, dipendente dalle sua caratteristiche statistiche. Tale ruolo è centrale e verrà mantenuto, come vedremo, anche nelle altre tipologie di codifica (*B-B*, *LV-B*, *LV-LV*). Ricordiamo inoltre che è possibile caratterizzare l'entropia anche per via algoritmica, sfruttando la *complessità di Kolmogorov* delle sequenze [19].

Il teorema di Shannon lascia però aperto il problema dell'individuazione del codice ottimo. Come vedremo nel seguito (sez. 5.2) si può dimostrare facilmente che il codice di Shannon-Fano introdotto nell'ambito del teorema 3.14 non è ottimo al finito. Osserviamo inoltre fin d'ora che, a norma del teorema 3.10, nel caso di una codifica *B-LV* la ricerca del codice ottimo può essere fatta limitandosi alla classe dei codici a prefisso.

3.4.8 Codifica *LV-B*

Nella codifica da lunghezza variabile a blocco la sequenza primaria uscente dalla sorgente $\mathcal{S}$ viene segmentata in una successione di messaggi a lunghezza variabile. A ciascun messaggio si associa una parola di codice a lunghezza costante. L'operazione viene eseguita avendo a disposizione una famiglia di messaggi esauriente $\mathcal{M}$ e in modo tale che qualunque sia la successione in uscita dalla $\mathcal{S}$ si riesca a trovare sempre (almeno) un prefisso appartenente a $\mathcal{M}$. Se la famiglia è a prefisso si evitano i ritardi di codifica. La richiesta di univoca decodificabilità viene invece soddisfatta in modo banale, non appena si scelgano parole di codice distinte per i diversi messaggi. Poiché le parole di codice sono a lunghezza costante è garantita anche l'istantaneità nella decodifica. Detta proprietà mette al riparo anche dai ritardi di codifica, che si possono però superare effettuando la segmentazione in corrispondenza del primo prefisso utile.

Sia dunque $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ l'alfabeto della sorgente $\mathcal{S}$ stazionaria e senza memoria e $P = \{p_1, p_2, \ldots, p_K\}$ la d.p. associata; si scelga inoltre una famiglia completa di messaggi $\mathcal{M}(j) = \{m_1, m_2, \ldots, m_{T(j)}\}$ derivante da j estensioni della $\mathcal{M}(0) = \mathcal{A}$. La cardinalità della famiglia vale $T(j) = K + j(K-1)$.

Se vogliamo disporre di almeno una parola di codice distinta per ogni messaggio deve valere la condizione

$$D^l \geq T(j)$$

che impone di scegliere come lunghezza costante per le parole di codice la quantità $l = \lceil \log_D T(j) \rceil$. Per la stazionarietà e l'assenza di memoria, se $\mathcal{N}(j) = \{n_1, n_2, \ldots, n_{T(j)}\}$ sono le lunghezze dei vari messaggi, si ha $Q(j) = \{q_1, q_2, \ldots, q_{T(j)}\}$, con $\Pr\{m_i\} = q_i = p_1^{i_1} p_2^{i_2} \ldots p_K^{i_K}$ e dove $\sum_{r=1}^{K} i_r = n_i$. L'efficienza della codifica viene misurata dal tasso (3.60), tenendo conto che $E[L] = l = \lceil \log_D T(j) \rceil$ è costante e dipende solo dalla cardinalità di $\mathcal{M}(j)$, cioè dal numero di estensioni fatte. Il tasso vale

$$R_j = \frac{l}{E[N_j]} = \frac{\lceil \log_D T(j) \rceil}{\sum_{i=1}^{T(j)} q_i n_i} \tag{3.69}$$

Nella codifica *LV-B* il parametro j gioca lo stesso ruolo del parametro asintotico n nel caso della codifica *B-LV*. Anche in questo caso verificheremo che l'entropia costituisce una barriera insuperabile oltre la quale non si può spingere alcuna compressione dei dati senza che venga meno l'univoca decodificabilità. Prima di dimostrare il teorema analogo a quello di Shannon per i codici *B-LV* abbiamo bisogno di un lemma preparatorio:

Lemma 3.3. *Se* $H_{_D}(Q(j)) = -\sum_{i=1}^{T(j)} q_i \log_{_D} q_i$ *è l'entropia associata alla d.p. dei messaggi di una famiglia completa, allora* $\forall j$ *vale la seguente relazione*

$$H_{_D}(Q(j)) = E[N_j]\, H_{_D}(P) \tag{3.70}$$

Dim. Si dimostra per induzione su j. Per $j = 0$ la (3.70) è vera, in quanto $E[N_0] = 1$ e $Q(0) = P$. Supponiamola vera per j e verifichiamone la validità per $j+1$. Si scelga un messaggio m^* di probabilità q^* per effettuare l'estensione dalla famiglia $\mathcal{M}(j)$ alla $\mathcal{M}(j+1)$ secondo la (3.58). Il legame tra le entropie al passo j e $j+1$ è dato da

$$\begin{aligned} H_{_D}(Q(j+1)) &= H_{_D}(Q(j)) + q^* \log_{_D} q^* - \sum_{i=1}^{K} (q^* p_i) \log_{_D} (q^* p_i) = \\ &= H_{_D}(Q(j)) - \sum_{i=1}^{K} q^* p_i \log_{_D} p_i = H_{_D}(Q(j)) + q^* H_{_D}(P) = \\ &= (E[N_j] + q^*)\, H_{_D}(P) \end{aligned}$$

dove l'ultima eguaglianza segue dall'applicazione dell'ipotesi induttiva. D'altra parte

$$E[N_{j+1}] = E[N_j] - q^* n^* + \sum_{i=1}^{K} q^* p_i\, (n^* + 1) = E[N_j] + q^* \tag{3.71}$$

che sostituita nella precedente porta alla tesi □

Possiamo ora affrontare il teorema centrale per la codifica *LV-B*

Teorema 3.16. *Il tasso di un qualunque codice LV-B a prefisso, per la* j*-esima estensione di una famiglia di messaggi associata a una sorgente* $\mathcal{S}$ *stazionaria e senza memoria, è inferiormente limitato dall'entropia* D*-aria* $H_{_D}(P)$ *della sorgente*.

Dim. Dalla definizione di tasso e dal lemma si ricava

$$R_j = \frac{l}{E[N_j]} = l \frac{H_{_D}(P)}{H_{_D}(Q(j))} \geq \lceil \log_{_D} T(j) \rceil \frac{H_{_D}(P)}{\log_{_D} T(j)} \geq H_{_D}(P) \tag{3.72}$$

dove la prima disuguaglianza deriva dal fatto che ogni entropia è limitata superiormente dal logaritmo della cardinalità. □

Anche in questo caso l'entropia è la limitazione inferiore per i tassi di qualunque codice a prefisso del tipo *LV-B*. Si osservi fin d'ora che nel caso della codifica *LV-B* manca l'analogo del teorema 3.10; ciò implica che la ricerca del codice ottimo deve essere estesa *a tutta la classe dei codici esaurienti*, e non solo alla classe dei codici a prefisso.

3.4.9 Codifica *LV-LV*

Il caso $LV-LV$ costituisce il paradigma più generale di codifica, che dovrebbe consentire, almeno in linea di principio, la massima flessibilità ed efficienza; in questo tipo di codifica si può agire infatti contemporaneamente tanto sulle lunghezze dei messaggi quanto su quelle relative alle parole di codice. Anche in questo caso l'entropia costituisce una limitazione inferiore per il tasso, come verificheremo fra poco.

Ogni codice *LV-LV* può essere pensato come concatenazione di un codice *LV-B* con un codice *B-LV*; infatti il flusso d'informazione che esce dalla sorgente deve essere segmentato sulla base di una famiglia completa di messaggi $\mathcal{M}(j) = \{m_1, m_2, \ldots, m_{_T}\}$, di lunghezze $\mathcal{N}(j) = \{n_1, n_2, \ldots, n_{_{T(j)}}\}$ e caratterizzati da una d.p. pari a $Q(j) = \{q_1, q_2, \ldots, q_{_{T(j)}}\}$. Ciascun messaggio deve poi essere codificato a lunghezza variabile, producendo un dizionario $W(j) = \{w_1, w_2, \ldots, w_{_{T(j)}}\}$ le cui lunghezze sono $L(j) = \{l_1, l_2, \ldots, l_{_{T(j)}}\}$. Naturalmente la scelta della struttura della famiglia $\mathcal{M}(j)$ determina l'effettiva prestazione del codice; infatti una volta fissata $\mathcal{M}(j)$ si determina implicitamente anche $Q(j)$, e a quel punto non resta altro da fare che minimare le lunghezze medie del codice $B-LV$.

Nell'estensione a lunghezza variabile di una famiglia completa possiamo usare la relazione (3.70) $H_{_D}(Q(j)) = E[N_j]\, H_{_D}(P)$, che lega l'entropia della sorgente con l'entropia delle foglie dell'albero completo del codice *LV-B*. Tenuto infine conto del teorema di Shannon 3.13 possiamo scrivere

$$E[L_j] \geq H_{_D}(Q(j)) = E[N_j]\, H_{_D}(P)$$

dal quale si ricava immediatamente il seguente

Teorema 3.17. *Il tasso di un arbitrario codice D-ario LV-LV su una famiglia completa $\mathcal{M}(j)$ è inferiormente limitato dall'entropia (D-aria) della sorgente*

$$R_j = \frac{E[L_j]}{E[N_j]} \geq H_{_D}(P) \tag{3.73}$$

3.5 Codifica a lunghezza costante

Le codifiche a lunghezza variabile precedentemente discusse presentano un inconveniente legato alla necessità di sincronizzare il funzionamento del codificatore con quello del canale. Se entrambi questi dispositivi funzionano con una cadenza temporale fissa e determinata dalla frequenza dell'orologio interno al sistema di elaborazione, accade che certe porzioni d'informazione possono andare perse.

Per fissare le idee si supponga che la sorgente emetta una lettera primaria per unità di tempo, e che nella stessa unità di tempo il canale sia in grado di accettare p.es. $E[L] = 5$ lettere secondarie. Se effettuiamo una codifica *B-LV* usando un codice efficiente, p.es. un codice di Shannon-Fano, dovremo attribuire alle lettere meno probabili delle parole di codice con una lunghezza maggiore della media $E[L]$. La comparsa consecutiva di alcune di queste lettere comporta un troncamento delle parole di codice in corrispondenza del ciclo di funzionamento successivo del canale, con una conseguente perdita d'informazione. Si rende allora necessario introdurre una *memoria tampone*, che consenta di salvaguardare l'informazione che il canale non è in grado di smaltire sul momento, e che verrà trasmessa in un tempo successivo, quando si presenteranno parole di codice più corte. D'altra parte un aumento del tasso di funzionamento del canale, che consenta allo stesso di accogliere anche le parole più lunghe, renderebbe inutile la compressione effettuata sui dati. Si noti che un problema analogo ricorre anche nello schema *LV-B*, nel quale si ha un'uscita del codificatore caratterizzata da una cadenza temporale irregolare. Anche se in pratica l'introduzione di una memoria tampone opportunamente dimensionata (associata a un dispositivo d'allarme che segnali l'intasamento del tampone) risolve il problema, dal punto di vista teorico una codifica da blocco a blocco (*B-B*) consentirebbe di evitarlo del tutto, poiché rimuoverebbe alla radice i problemi di sincronizzazione.

Analizziamo dunque quest'ultimo caso. Se al solito $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ è l'alfabeto primario della sorgente e $\mathcal{B} = \{b_1, b_2, \ldots, b_D\}$ è quello secondario, effettuiamo una codifica dell'estensione n-ima della sorgente. A ciascun blocco primario di lunghezza n si associ un blocco secondario di lunghezza l. Per garantire l'univoca decodificabilità è necessario che esista almeno un blocco secondario per ciascuna n-pla primaria di $\mathcal{A}^n$, cioè

$$D^l \geq K^n \qquad \text{e dunque} \qquad l \geq n \log_D K \tag{3.74}$$

Per rendere minimo il tasso, l dev'essere il più piccolo intero compatibile con la (3.74); prendiamo allora

$$l = \lceil n \log_D K \rceil \tag{3.75}$$

Ciò comporta che per il tasso valgano le seguenti relazioni

$$R_n = \frac{\lceil n \log_D K \rceil}{n} \geq \log_D K \geq H_D(P)$$
$$R_n < \log_D K + \frac{1}{n} \tag{3.76}$$

e dunque, a meno che la distribuzione P non sia uniforme ($P \equiv U$, e in questo caso il problema della compressione dei dati non si pone), il tasso risulta sempre *strettamente maggiore* della limitazione *superiore* per l'entropia della sorgente (si veda l'oss. 3.7). Non è dunque possibile sperare nell'esistenza di (successioni di) codici il cui tasso si approssimi asintoticamente all'entropia, così come succede per i codici a lunghezza variabile. Di conseguenza i codici *B-B* non sono in grado di comprimere i dati, ma effettuano una semplice traduzione da un alfabeto a un altro.

Tutto ciò è conseguenza della richiesta di univoca decodificabilità. Rimuovendola si potrebbe diminuire l, a parità di n, usando un insieme di parole di codice con cardinalità D^l minore di K^n. Così facendo non siamo in grado di attribuire parole di codice distinte a *tutte* le n-ple primarie; ciò può essere tollerabile solo nel caso in cui le n-ple non distinguibili cumulino una probabilità d'uscita molto bassa. A queste ultime si può attribuire una qualunque parola di codice, e quando una di queste viene emessa dalla sorgente si verifica un *errore di decodifica*, essendo nell'impossibilità di decodificare correttamente.

Vediamo di formalizzare meglio il problema. Il codice *B-B* è caratterizzato da un *funzione di codifica* φ

$$\varphi : \mathcal{A}^n \to \mathcal{B}^l \tag{3.77}$$

e da una *funzione di decodifica* ψ

$$\psi : \mathcal{B}^l \to \mathcal{A}^n \tag{3.78}$$

Fissare un codice significa fissare una coppia φ, ψ. Se $D^l < K^n$ si possono verificare degli errori in decodifica; ciò accade quando la decodifica della codifica di un messaggio primario $\boldsymbol{x}$ è diversa da $\boldsymbol{x}$. La probabilità d'errore cumulata risulta pari a

$$P_{err} \triangleq \Pr\{\boldsymbol{x} : \psi[\varphi(\boldsymbol{x})] \neq \boldsymbol{x}\} \tag{3.79}$$

Viceversa, quando $\psi[\varphi(\boldsymbol{x})] = \boldsymbol{x}$ non si commette alcun errore. Consideriamo allora il sottoinsieme $\mathcal{T}$ di $\mathcal{A}^n$ definito come

$$\mathcal{T} = \{\boldsymbol{x} : \psi[\varphi(\boldsymbol{x})] = \boldsymbol{x}\} \tag{3.80}$$

Dunque tutte le n-ple di $\mathcal{T}$ hanno parole di codice distinte, e naturalmente $|\mathcal{T}| \leq D^l$. Nel codice possiamo creare una corrispondenza biunivoca fra tutte le n-ple

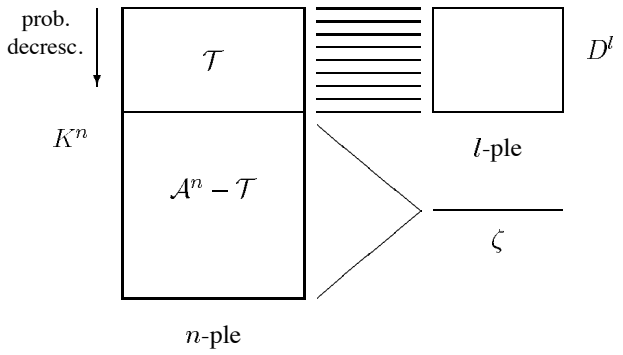


Figura 3.4 Insiemi implicati nella codifica B-B.

di $\mathcal{T} \subset \mathcal{A}^n$ e le D^l l-ple. Per la codifica delle n-ple di $\mathcal{A}^n - \mathcal{T}$ possiamo usare le parole di codice già usate per $\mathcal{T}$, oppure possiamo fissare una parola *ambigua* $\boldsymbol{\zeta}$, diversa dalle altre, che viene impiegata per codificare un qualunque messaggio di $\mathcal{A}^n - \mathcal{T}$, e la cui ricezione comporta un errore di decodifica (vedi fig.3.4). La decodifica avviene secondo i seguenti schemi

- se usiamo sempre le stesse parole tanto per $\mathcal{T}$ che per $\mathcal{A}^n - \mathcal{T}$, possiamo imporre

  $$\psi[\varphi(\boldsymbol{x})] \in \mathcal{T} \qquad \forall \boldsymbol{x}$$

  Dunque, se $\mathcal{S}$ emette una n-pla di $\mathcal{T}$ la decodifica è corretta, altrimenti commettiamo errore;

- se usiamo la parola ambigua, detta $\boldsymbol{y}$ la parola di codice ricevuta se $\boldsymbol{y} \neq \boldsymbol{\zeta}$ si decodifica normalmente come $\boldsymbol{x} = \psi(\boldsymbol{y})$, altrimenti si decodifica a piacere commettendo errore.

Poiché $D^l \geq |\mathcal{T}|$ scegliamo al solito il più piccolo l compatibile con la disuguaglianza, cioè $l = \lceil \log_D |\mathcal{T}| \rceil$. I parametri che definiscono il codice sono allora la *probabilità d'errore*, associata alla probabilità che esca una n-pla di $\mathcal{A}^n - \mathcal{T}$, e il *tasso*, definito nei termini della cardinalità dell'insieme $\mathcal{T}$

$$\begin{aligned} P_{err}^n &= P^n(\mathcal{A}^n - \mathcal{T}) = 1 - P^n(\mathcal{T}) \\ R_n &= \frac{\lceil \log_D |\mathcal{T}| \rceil}{n} \end{aligned} \tag{3.81}$$

Poiché D e n sono fissati, l'elemento cruciale nella costruzione del codice è l'insieme $\mathcal{T}$. Il problema da risolvere nella costruzione di un buon codice blocco è allora il seguente:

Codifica *B-B*. Individuare, se esiste, un insieme $\mathcal{T}$ (una successione di insiemi $\mathcal{T}^{(n)}$) che possegga una probabilità elevata con una cardinalità bassa.

Ragioneremo nel seguente modo: fissato come *parametro di progetto* ϵ, la massima probabilità d'errore tollerata, cerchiamo un insieme $\mathcal{T}$ tale che $P_{err}^{n} = 1 - P^{n}(\mathcal{T}) \leq \epsilon$, cioè $P^{n}(\mathcal{T}) \geq 1 - \epsilon$. Si giunge allora alla definizione del seguente

Codice ottimo *B-B*. Si ordinano tutte le n-ple per probabilità decrescenti. Si prende $\boldsymbol{\alpha_1}$, la prima della lista (la più probabile), e la si mette in $\mathcal{T}$, verificando se $P^{n}(\mathcal{T}) \geq 1 - \epsilon$. Se la relazione non è verificata si pone anche $\boldsymbol{\alpha_2}$ in $\mathcal{T}$, procedendo a una nuova verifica e iterando il procedimento fin quando la verifica dà esito positivo.

L'insieme $|\mathcal{T}|$ così costruito è il *minimo insieme* che cumula una probabilità globale maggiore di $1 - \epsilon$, e di conseguenza il codice che se ne ricava è ottimo, poichè il corrispondente tasso è minimo.

L'algoritmo appena descritto ci consente però di valutare le prestazioni del codice solo in linea di principio, cioè costruendo materialmente l'insieme $\mathcal{T}$ a partire dalla d.p. della $\mathcal{S}$. Non siamo inoltre in grado di fare delle previsioni sul comportamento asintotico del tasso e capire se esso si può avvicinare o meno all'entropia di $\mathcal{S}$. Per risolvere questo problema introdurremo dei codici sub-ottimali, ma dotati di una struttura che ci consentirà di calcolare il tasso. Le prestazioni ricavate per detti codici saranno le minime garantite per il codice ottimo. Poiché l'obiettivo è quello di individuare insiemi a bassa cardinalità ed elevata probabilità, possiamo far riferimento agli insiemi di n-ple *tipiche*, in probabilità e in composizione, di cui si è parlato nella sezione 3.3. Per tali insiemi vale la proprietà di *equipartizione asintotica*: al tendere di n all'infinito le n-ple generate si dividono in due insiemi: il primo, quello delle *sequenze tipiche*, contiene *pochi* elementi (circa $D^{nH_D(P)}$) a probabilità quasi uniforme (circa $D^{-nH_D(P)}$) e cumula una probabilità complessiva che tende a 1 quando $n \to \infty$. Il secondo, quello delle sequenze *non tipiche*, contiene *quasi tutte* le sequenze ($D^{n \log_D K} - D^{nH_D(P)} \approx D^{n \log_D K}$), ma cumula una probabilità asintoticamente nulla. Facendo riferimento alla tipicità in probabilità, possiamo riprendere le (3.26) e (3.27)

$$\begin{aligned} &\mathcal{T}_{\mathcal{P}}^{(n,\delta)} = \left\{ \boldsymbol{x} : \left| \frac{1}{n} \mathcal{I}(\boldsymbol{x}) - H_D(P) \right| \leq \delta \right\} \\ &\Pr\left\{ \mathcal{T}_{\mathcal{P}}^{(n,\delta)} \right\} \geq 1 - \epsilon \\ &(1-\epsilon)\, D^{n[H_D(P) - \delta]} \leq \left| \mathcal{T}_{\mathcal{P}}^{(n,\delta)} \right| \leq D^{n[H_D(P) + \delta]} \end{aligned} \tag{3.82}$$

Scegliendo

$$D^l \geq D^{n[H_D(P)+\delta]} \geq \left|\mathcal{T}_p^{(n,\delta)}\right|$$

cioè $R_n \geq H_D(P)+\delta$, si può attribuire una parola di codice a ciascuna sequenza tipica. Con questa scelta, a norma della (3.27), la probabilità d'errore rimane limitata superiormente da ϵ, che è infinitesimo. Viceversa, se scegliamo un tasso R_n strettamente inferiore all'entropia, la probabilità d'errore tende asintoticamente a 1; infatti ponendo $R_n \leq H_D(P) - 2\delta$, cioè $D^l \leq D^{n[H_D(P)-2\delta]}$ e tenendo conto della (3.23) (cioè che la probabilità di ogni sequenza tipica è al più $D^{-n[H_D(P)-\delta]}$), la probabilità globale cumulata dalle sequenze per le quali esistono parole di codice vale al più

$$D^{-n[H_D(P)-\delta]} \cdot D^{n[H_D(P)-2\delta]} = D^{-n\delta}$$

Anche attribuendo alcune parole di codice a sequenze esterne a $\mathcal{T}_p^{(n,\delta)}$, p.es. quelle a probabilità più elevata, sappiamo che esse cumulano una probabilità che vale comunque al più ϵ. La probabilità di tutte le sequenze per le quali si attribuisce una parola di codice, appartenenti o meno all'insieme delle sequenze tipiche, è dunque limitata superiormente da

$$1 - P_{err}^n \leq \epsilon + D^{-n\delta}$$

Così quando $n \to \infty$, con $R_n \leq H_D(P) - 2\delta$, si ha $P_{err}^n \to 1$. Ciò consente di concludere con il celebre

Teorema 3.18 (*B-B* di Shannon).

(Parte diretta) - Se $R_n \geq H_D(P)+\delta$, $\forall\delta > 0$ *esiste una successione di codici B-B per le estensioni* n*-esime della sorgente* $\mathcal{S}$ *per i quali* $\lim_{n\to\infty} P_{err}^n = 0$

(Parte inversa) - Se $R_n \leq H_D(P) - 2\delta$ *allora* $\lim_{n\to\infty} P_{err}^n = 1$

Poiché la sorgente genera asintoticamente "solo" sequenze tipiche, il procedimento appena descritto consiste nel codificare, tra tutte le possibili sequenze, solo queste ultime. Tutte le altre vanno trascurate, e ciò comporta una certa probabilità d'errore, che però può essere limitata da un infinitesimo quando n tende all'infinito. La parte inversa del teorema stabilisce invece l'impossibilità di scendere col tasso sotto l'entropia, e ciò *anche accontentandosi di una probabilità d'errore non asintoticamente nulla*; infatti, l'ipotesi di un tasso sotto l'entropia comporta una probabilità d'errore asintoticamente unitaria.

Per evitare di trascurare le sequenze non tipiche, si può attribuire loro un numero sufficiente di parole di codice a lunghezza costante, il che consente di distinguerle l'una dall'altra evitando così ogni errore. Ciò porta alla costruzione del seguente codice univocamente decodificabile

Codice a due lunghezze. Alle sequenze tipiche, che sono al più $2^{n(H_D(P)+\delta)}$, si assegnino parole di codice di lunghezza $l_T = \lceil n(H_D(P)+\delta)\rceil$. A ciascuna di esse si dia prefisso 1. A tutte le sequenze non tipiche, che sono almeno $K^n - 2^{n(H_D(P)+\delta)}$, si assegnino invece parole di codice di lunghezza $l_{NT} = \lceil n \log K\rceil$, dando come prefisso 0.

Si può verificare che la codifica delle sequenze non tipiche non comporta un peggioramento delle prestazioni del codice in termini di tasso, poiché il contributo alla lunghezza media di queste sequenze è asintoticamente nullo.

Proposizione 3.3. *Il tasso per un codice a due lunghezze tende asintoticamente all'entropia della sorgente.*

Dim. La presenza del prefisso 0 o 1 consente di discriminare tra parole di codice per sequenze tipiche e non tipiche. Le due lunghezze effettive sono dunque $l_T + 1$ e $l_{NT} + 1$. Calcoliamo la lunghezza media

$$\begin{aligned} E[L^n] &= \sum_{\boldsymbol{x}\in\mathcal{X}^n} p(\boldsymbol{x})l(\boldsymbol{x}) = \sum_{\boldsymbol{x}\in\mathcal{T}_{\mathcal{P}}^{(n,\delta)}} p(\boldsymbol{x})l(\boldsymbol{x}) + \sum_{\boldsymbol{x}\in\left\{\mathcal{X}^n - \mathcal{T}_{\mathcal{P}}^{(n,\delta)}\right\}} p(\boldsymbol{x})l(\boldsymbol{x}) = \\ &= (l_T+1)\Pr\{\mathcal{T}_{\mathcal{P}}^{(n,\delta)}\} + (l_{NT}+1)\left(1 - \Pr\{\mathcal{T}_{\mathcal{P}}^{(n,\delta)}\}\right) = \\ &= \left(\lceil n(H_D(P)+\delta)\rceil + 1\right)(1-\epsilon_{n,\delta}) + \left(\lceil n\log K\rceil + 1\right)\epsilon_{n,\delta} < \\ &< n(H_D(P)+\delta) + 2 + (n\log K + 2)\,\epsilon_{n,\delta} \end{aligned} \tag{3.83}$$

Dividendo per n si ha l'espressione del tasso n-esimo

$$R_n = \frac{E[L^n]}{n} < H_D(P) + \delta + \beta_n \tag{3.84}$$

con $\beta_n = \frac{2}{n} + \left(\log K + \frac{2}{n}\right)\epsilon_{n,\delta} \to 0$, infinitesimo quando $n \to \infty$ □

Effettuando una codifica asintotica *B-LV* (p.es alla Shannon-Fano) per *tutte* le sequenze, alle tipiche viene attribuita una lunghezza pari a $\lceil -\log p\rceil \approx nH_D(P)$, cioè circa costante e uguale alla lunghezza attribuita dal codice a due lunghezze; le non tipiche vengono invece trattate diversamente, poiché nel caso di Shannon-Fano hanno lunghezza variabile. Ciò consente di migliorare la prestazione offerta del codice a due lunghezze, visto che per il codice di Shannon-Fano si ha $R_n < H_D(P) + \frac{1}{n}$. Tale differenza si annulla però asintoticamente, poiché il contributo β_n, che deriva alla lunghezza media dalle sequenze non tipiche, è asintoticamente nullo.

Capitolo 4
Codifica di canale

4.1 Introduzione

Inizieremo ora lo studio del secondo grosso capitolo della teoria di Shannon, quello riguardante le tecniche per la protezione dal rumore del segnale che transita sul canale. Il canale è il sostegno fisico che consente il passaggio del flusso d'informazione dalla sorgente all'utente. Nelle telecomunicazioni esso viene identificato normalmente con una linea elettromagnetica, p.es. un doppino telefonico, un cavo coassiale, una guida d'onda, una fibra ottica o lo stesso "etere", nel caso delle trasmissioni per irradiazione diretta di campi elettromagnetici. In ciascuno di questi sostrati fisici è sempre presente una certa quantità di *rumore*, che si somma al segnale che viene trasmesso. Esso è la sovrapposizione di un insieme molto grande di disturbi di varia natura, il cui effetto medio può essere ricondotto, almeno per i sistemi discreti, alla definizione di un'opportuna *probabilità d'errore* ϵ . Ciò significa che, se usiamo all'ingresso del canale un alfabeto q-ario $x_1, x_2, \ldots, x_q$ e trasmettiamo sullo stesso una lettera x_i, con probabilità $\epsilon > 0$ troviamo in uscita $x_j \neq x_i$. Il valore di ϵ dipende in modo rilevante dal tipo di canale, anche se da un punto di vista matematico l'unica cosa che interessa è sapere che trattasi di numero reale compreso tra zero e uno.

Le telecomunicazioni non forniscono però l'unico esempio di impiego delle tecniche che illustreremo nel seguito, ma solamente quello più frequente o più noto. Anche tutte le operazioni di trascrizione dell'informazione da un sostrato fisico all'altro sono di norma affette da una certa imprecisione, e anche questa può essere descritta nei termini statistici di una probabilità d'errore; si pensi, p.es. alla lettura laser dei bit dislocati su un disco ottico (CD), e alla possibilità che imperfezioni della superficie del disco, o anche solo impurità depositate sulla stessa, possano pregiudicare la corretta interpretazione dei bit. Un discorso analogo vale per la memorizzazione su un supporto magnetico, quale nastro o disco.

Le tecniche che apprenderemo nel seguito sono pertanto tese a individuare gli errori che dovessero verificarsi, ed eventualmente a correggerli. Anche se da un

punto di vista tecnico-applicativo questo sembra essere il contributo più rilevante derivante dalla teoria di Shannon, non si deve dimenticare che c'è anche una parte normativa importantissima, che determina le prestazioni limite, al finito e asintotiche, dei sistemi affetti da rumore. È questa la parte concettualmente più ricca e profonda della teoria dei codici di canale, e non mancheremo di analizzarla in modo dettagliato nel corso di questo terzo capitolo. Lasceremo invece alla seconda parte la trattazione sistematica dei codici correttori, la loro costruzione e la valutazione delle prestazioni offerte in termini di tasso di trasmissione e di capacità di correzione del codice. Come vedremo queste si discostano ancora in modo significativo da quelle ottime promesse dai teoremi di codifica.

Consideriamo ora il sistema di comunicazione illustrato nella figura 4.1. Per il momento fissiamo l'ipotesi che l'alfabeto d'ingresso dal canale sia binario ($q = 2$), anche se ciò non pregiudica la generalità di quanto diremo. Supponiamo che la sorgente emetta una lettera binaria u appartenente all'alfabeto $\mathcal{U} = \{0, 1\}$ e che tale lettera si presenti all'ingresso del canale, caratterizzato da due simboli d'ingresso associati al *dizionario* $\mathcal{X} = \{0, 1\}$. Consideriamo il modello di *il canale simmetrico binario* (vedi par. 4.2.5), nel quale ciascuna lettera d'ingresso può essere modificata con una probabilità $\epsilon > 0$. La lettera in uscita al canale appartiene all'insieme $\mathcal{Y} = \{0, 1\}$. Sulla base di quanto ricevuto si effettua allora una *stima* $v = \widehat{u}$ di ciò che è stato trasmesso, usando l'insieme $\mathcal{V} = \{0, 1\}$. Se $\widehat{u} \neq u$ allora si è verificato un *errore*. Se dopo aver introdotto

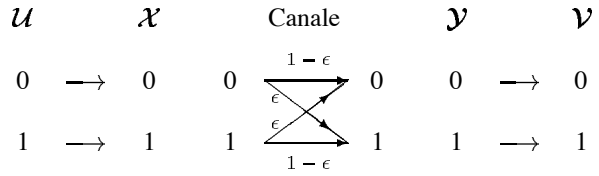


Figura 4.1 Sistema di comunicazione non protetto dal rumore.

"0" all'ingresso del canale riceviamo "1" in uscita non siamo in grado di accorgerci dell'errore, poiché anche "1" è una lettera che può legittimamente essere stata trasmessa, in quanto $1 \in \mathcal{X}$. Lo stesso dicasi trasmettendo "1" e ricevendo "0". Supponendo che le v.a. U, X, Y, V prendano i loro valori sui corrispondenti insiemi $\mathcal{U}, \mathcal{X}, \mathcal{Y}, \mathcal{V}$, calcoliamo la probabilità d'errore, cioè la probabilità che sia $U \neq V$

$$\begin{aligned} P_e &= \Pr\{U \neq V\} = \Pr\{U = 0, V = 1\} + \Pr\{U = 1, V = 0\} = \\ &= \Pr\{X = 0, Y = 1\} + \Pr\{X = 1, Y = 0\} = \\ &= \Pr\{X = 0\}\Pr\{Y = 1/X = 0\} + \Pr\{X = 1\}\Pr\{Y = 0/X = 1\} \\ &= \epsilon\left[\Pr\{X = 0\} + \Pr\{X = 1\}\right] = \epsilon \end{aligned} \tag{4.1}$$

dove l'ultima uguaglianza segue dal fatto che $\Pr\{Y = 0/X = 1\} = \Pr\{Y = 1/X = 0\} = \epsilon$ a seguito della struttura del canale. Anche se ϵ è abbastanza piccolo, dopo aver trasmesso n lettere consecutive (nelle ipotesi di stazionarietà e assenza di memoria del processo) con n sufficientemente grande si ha $n\epsilon > 1$, e dunque si verificherà almeno un errore.

Si può tentare di risolvere il problema aumentando la lunghezza della sequenza che codifica il simbolo u generato; se usiamo sequenze di lunghezza ≥ 2, si può infatti escogitare un metodo per rendersi conto degli errori, ed eventualmente correggerli, in modo da abbattere il valore di P_e specificato dalla (4.1).

Supponiamo di prendere $n = 2$; le sequenze possibili sono le quattro coppie binarie $00, 01, 10, 11$. Effettuiamo una codifica dei bit che escono dalla sorgente nel modo seguente: quando esce $u = 0$ sul canale si manda la coppia 00, quando $u = 1$ si manda la coppia 11; 00 e 11 costituiscono le *parole di codice*. L'operazione effettuata corrisponde all'immissione di *ridondanza strutturale e sistematica* nell'informazione che esce dalla sorgente, e ciò consente di rendersi conto della presenza di un errore. Se infatti il primo bit viene modificato per effetto del rumore, uno 00 trasmesso diventa 10 ricevuto, e poiché quest'ultima coppia *non corrisponde ad alcuna parola di codice* ci si accorge della presenza di un errore.

Il procedimento può essere migliorato usando $n = 3$ e tentando una correzione dell'errore mediante una decodifica *a maggioranza*, così come riportato nello schema dell'esempio seguente.

Esempio 4.1. Quando esce "0" o "1", il codificatore introduce ridondanza *ripetendo per tre volte* il simbolo. Sul canale viene dunque immessa la terna "000" o "111". A causa del rumore, in ricezione può capitare di trovare una qualunque tra le 2^3 n-ple possibili, e la decodifica viene fatta *a maggioranza*, cioè decidendo "0" se c'è una maggioranza di zeri o "1" se c'è maggioranza di uni. Calcoliamo la probabilità d'errore associata a questo schema, nelle ipotesi di stazionarietà e di assenza di memoria per il canale simmetrico binario , tenuto conto che in un blocco di lunghezza n la probabilità di avere w errori è pari a $\epsilon^w(1-\epsilon)^{n-w}$ e dipende solo da w

$$\begin{aligned}
\Pr\{V = 1/U = 0\} &= \Pr\{Y = 111/X = 000\} + \Pr\{Y = 110/X = 000\} + \\
&+ \Pr\{Y = 101/X = 000\} + \Pr\{Y = 011/X = 000\} \\
&= 3\epsilon^2(1-\epsilon\} + \epsilon^3 \\
\Pr\{V = 0/U = 1\} &= \Pr\{Y = 000/X = 111\} + \Pr\{Y = 001/X = 111\} \\
&+ \Pr\{Y = 010/X = 111\} + \Pr\{Y = 100/X = 111\} \\
&= 3\epsilon^2(1-\epsilon) + \epsilon^3
\end{aligned}$$

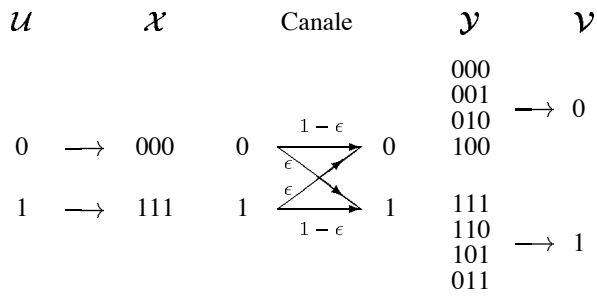


Figura 4.2 Sistema di comunicazione con decodifica a maggioranza.

e dunque

$$\begin{aligned} P_e &= \Pr\{U \neq V\} = \Pr\{U=0, V=1\} + \Pr\{U=1, V=0\} = \\ &= \Pr\{U=0\}\Pr\{V=1/U=0\} + \Pr\{U=1\}\Pr\{V=0/U=1\} \\ &= 3\epsilon^2(1-\epsilon) + \epsilon^3 \end{aligned} \tag{4.2}$$

Si noti che per $\epsilon < 0.5$ si ha $3\epsilon^2(1-\epsilon) + \epsilon^3 < \epsilon$, e dunque si riesce a diminuire la probabilità d'errore rispetto al caso con $n = 1$. ◯

In generale, assegnato un alfabeto q-ario $\mathcal{A} = \{a_1, a_2, \ldots, a_q\}$ avremo a che fare con tutte le possibili q^n stringhe di lunghezza n, che indichiamo col simbolo $\boldsymbol{q^n}$.

Def. 4.1. *Si definisce codice blocco q-ario o codice di canale un qualunque sottinsieme $\mathfrak{C} \subseteq \boldsymbol{q^n}$; ogni stringa $\boldsymbol{x}$ di $\mathfrak{C}$ si chiama parola di codice.*

Se $M = |\mathfrak{C}|$ è la cardinalità del codice, per poter riconoscere la presenza di un errore dovremo usare un numero

$$M < q^n \tag{4.3}$$

di parole di codice, a fronte delle q^n potenzialmente disponibili sull'alfabeto q-ario di canale. Se definiamo allora *tasso di trasmissione del codice* la quantità

$$R \triangleq \frac{1}{n} \log_q M \tag{4.4}$$

sussiste la relazione

$$R = \frac{1}{n} \log_q M \leq 1 \tag{4.5}$$

nella quale la disuguaglianza stretta esprime la presenza di ridondanza. Se n è la lunghezza delle parole di codice, possiamo pensare che di questi n q-it solo $\log_q M = k < n$ siano d'informazione, mentre i restanti $n-k$ siano di *controllo*. Con k q-it d'informazione si possono costruire $M = q^k$ parole di codice, e il tasso vale

$$R = \frac{\log_q q^k}{n} = \frac{k}{n}$$

A norma della (4.4) possiamo esprimere la cardinalità del dizionario del codice come

$$M = q^{nR} \tag{4.6}$$

Prima di studiare in modo sistematico le tecniche di rilevazione e correzione degli errori, presentiamo un modello di funzionamento del canale.

4.2 Canali e capacità

Anche per il modello del canale di trasmissione faremo le stesse ipotesi semplificative viste per la sorgente, cioè di *stazionarietà e di assenza di memoria* (per modelli più generali si veda [3]). Ciò significa che il comportamento stocastico del canale non risente di traslazioni temporali e non è influenzato da ciò che si è verificato nel passato del suo funzionamento. Un canale finito, stazionario e senza memoria (*CSSM*) è caratterizzato da un alfabeto d'ingresso $\mathcal{X} = \{x_1, x_2, \ldots, x_{|\mathcal{X}|}\}$ e da un alfabeto di uscita $\mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$. Esso opera su un insieme discreto di istanti di tempo $\mathfrak{T} = \{\ldots, t_0, t_1, t_2, \ldots\}$; in ogni istante $t \in \mathfrak{T}$ viene immessa una lettera $x_i \in \mathcal{X}$ in ingresso al canale e viene emessa una lettera $y_j \in \mathcal{Y}$ in uscita. Per effetto del rumore in uscita si può trovare, per ogni x_i d'ingresso, una qualunque lettera y_j. La situazione viene descritta matematicamente da una famiglia di $|\mathcal{X}|$ d.p. condizionate, che possono essere raccolte nella seguente *matrice di transizione*

$$\boldsymbol{\Gamma} = \begin{pmatrix} p(y_1/x_1) & p(y_2/x_1) & \ldots & p(y_{|\mathcal{Y}|}/x_1) \\ p(y_1/x_2) & p(y_2/x_2) & \ldots & \vdots \\ \vdots & \vdots & & \vdots \\ p(y_1/x_{|\mathcal{X}|}) & p(y_2/x_{|\mathcal{X}|}) & \ldots & p(y_{|\mathcal{Y}|}/x_{|\mathcal{X}|}) \end{pmatrix} \tag{4.7}$$

$$p(y_j/x_i) \geq 0 \quad \sum_{j=1}^{|\mathcal{Y}|} p(y_j/x_i) = 1$$

La stazionarietà e l'assenza di memoria è stabilita dalla relazione

$$\Pr\{Y_{k+1}\dots Y_{k+n} = y_{j_1}\dots y_{j_n}/X_{k+1}\dots X_{k+n} = x_{i_1}\dots x_{i_n}\} = \\ = \prod_{r=1}^{n} \Pr\{Y_{k+r} = y_{j_r}/X_{k+r} = x_{i_r}\} \tag{4.8}$$

cioè, semplificando la notazione

$$p(\boldsymbol{y}/\boldsymbol{x}) = \prod_{r=1}^{n} p(y_{j_r}/x_{i_r}) \tag{4.9}$$

La quantità media d'informazione scambiata nel canale tra le due variabili X e Y è pari alla mutua informazione delle stesse (vedi formula (2.5)), cioè

$$I(X \wedge Y) = \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p(x)p(y/x)\log\frac{p(y/x)}{p(y)}$$

dove $p(y) = \sum_{x\in\mathcal{X}} p(x,y) = \sum_{x\in\mathcal{X}} p(x)p(y/x)$, cioè

$$I(X \wedge Y) = \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p(x)p(y/x)\log\frac{p(y/x)}{\sum_{x\in\mathcal{X}} p(x)p(y/x)}$$

Una volta fissata la matrice di transizione $\boldsymbol{\Gamma}$, la mutua informazione funzione della sola d.p. d'ingresso. Inoltre, variando $p(x)$ varia $I(X \wedge Y)$, e ha dunque senso calcolare il massimo della mutua informazione *su tutte le distribuzioni di probabilità* assegnate all'ingresso del canale.

Def. 4.2. *Si definisce capacità del canale la quantità*

$$C \triangleq \max_{P} I(X \wedge Y) = \max_{P}[H(X) - H(X/Y)] = \\ = \max_{P}[H(Y) - H(Y/X)] \tag{4.10}$$

Dal punto di vista euristico la capacità può essere interpretata come la *massima quantità d'informazione* che il canale consente di scambiare alle due v.a. X e Y, ed è chiaro che tale informazione dipende, oltre che dalle caratteristiche fisiche di rumore riconducibili alla matrice $\boldsymbol{\Gamma}$, anche dalla d.p. d'ingresso P. La capacità ha inoltre un significato operativo centrale nella codifica di canale; come vedremo essa rappresenta una limitazione superiore per il tasso di tutti i codici correttori (con probabilità d'errore massimale limitata da una costante strettamente minore di 1).

Le entropie condizionate $H(X/Y)$ e $H(Y/X)$ si chiamano rispettivamente *equivocazione* e *ambiguità* . La prima rappresenta l'incertezza che rimane su X

noto Y, e si può dunque interpretare come l'informazione "persa" nel tragitto sul canale; la seconda rappresenta invece una quantità d'informazione che perviene a Y, ma che *non* deriva da X. Tuttavia il calcolo effettivo della capacità è reso complesso dai vincoli imposti per P, cioè $p(x) \geq 0$ e $\sum p(x) = 1$. Studieremo ora in modo sistematico alcuni tipi di canali per i quali il calcolo della capacità risulta particolarmente semplice. Per un algoritmo generale per il calcolo della capacità si veda [3] o [36].

4.2.1 Canale senza rumore

In questo caso si realizza una corrispondenza biunivoca tra ingresso e uscita. Se $|\mathcal{X}| = |\mathcal{Y}| = m$, a ciascun x_i corrisponde deterministicamente un y_i. La matrice $\boldsymbol{\Gamma}$ è identica, e per quanto attiene alle probabilità condizionate si ha

$$p(y_j/x_i) = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases}$$

da cui si ottiene $p(y_j) = \sum_i p(x_i)p(y_j/x_i) = p(x_j)$ e anche

$$p(x_i/y_j) = \frac{p(x_i, y_j)}{p(y_j)} = \frac{p(x_i)p(y_j/x_i)}{p(y_j)} = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases}$$

Ciò implica che tanto l'ambiguità che l'equivocazione sono nulle, e che nota X non si ha alcuna incertezza media su Y e viceversa. Per la mutua informazione si ottiene $I(X \wedge Y) = H(X) = H(Y)$. La massimazione sulla d.p. d'ingresso porta a

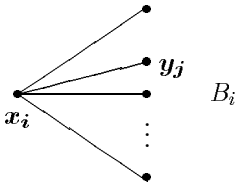
$$C = \max_P I(X \wedge Y) = \max_P H(X) = \log m \tag{4.11}$$

e la d.p. che realizza la capacità è quella uniforme.

4.2.2 Canale senza perdite

Un canale è senza perdite se l'ingresso X è completamente determinato dall'uscita Y. Più precisamente sia $|\mathcal{X}| = m$ e $|\mathcal{Y}| = s > m$; qualunque sia la distribuzione di probabilità d'ingresso, esiste una partizione $B_1, B_2, \ldots, B_m$ di $\mathcal{Y}$ ($B_i \cap B_j = \emptyset$ se $i \neq j$) tale che Ciò implica che $H(X/Y) = 0$ e l'equivocazione è nulla. Sfruttando questo fatto il calcolo della capacità si riconduce alla (4.11) del caso precedente, che porta ancora a un valore di $\log m$ in corrispondenza di una d.p. d'ingresso uniforme. Il canale senza perdite è dunque assimilabile, dal punto di vista sostanziale, a uno senza rumore, poiché non c'è degradazione dell'informazione ; nota l'uscita si risale infatti immediatamente all'ingresso.

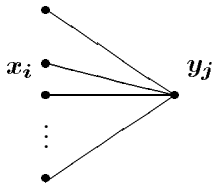
$$p(x_i/y_j) = \begin{cases} 1 & \text{se } y_j \in B_i \\ 0 & \text{se } y_j \notin B_i \end{cases}$$



4.2.3 Canale deterministico

Un canale è deterministico se l'uscita Y è completamente determinato dall'ingresso X. Più precisamente sia $|\mathcal{Y}| = m$ e $|\mathcal{X}| = s > m$; qualunque sia la distribuzione di probabilità d'uscita, esiste una partizione $A_1, A_2, \ldots, A_m$ di $\mathcal{X}$ $(A_i \cap A_j = \emptyset$ se $i \neq j)$ tale che In questo caso $H(Y/X) = 0$ e

$$p(y_j/x_i) = \begin{cases} 1 & \text{se } x_i \in A_j \\ 0 & \text{se } x_i \notin A_j \end{cases}$$



l'ambiguità è nulla. Con gli stessi calcoli di prima si ottiene

$$C = \max_P I(X \wedge Y) = \max_P H(Y) = \log m$$

La massimazione della mutua informazione porta in questa circostanza a una d.p. uniforme sull'uscita, cioè sull'insieme $\mathcal{Y}$. Per individuare la struttura della d.p. d'ingresso corrispondente è necessario risolvere un sistema di m equazioni in s incognite

$$p(y_j) = \sum_{i=1}^{s} p(x_i)p(y_j/x_i) = 1/m,\ 1 \leq j \leq m$$

cioè

$$\sum_{x_i \in A_j} p(x_i) = 1/m$$

4.2.4 Canale inutile

Nel canale inutile la matrice $\boldsymbol{\Gamma}$ ha le righe tutte uguali

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_m \\ \alpha_1 & \alpha_2 & \ldots & \alpha_m \\ & \vdots & & \\ \alpha_1 & \alpha_2 & \ldots & \alpha_m \end{pmatrix}$$

dove $\alpha_j \geq 0, \sum_{j=1}^{m} \alpha_j = 1$, essendo $\alpha_j = p(y_j/x_i) = p(y_j/x_k)$ con $i \neq k$. Ciò implica $p(y_j) = \sum_{i=1}^{m} p(x_i)p(y_j/x_i) = \alpha_j = p(y_j/x_i)$ e dunque X e Y sono indipendenti. In questo caso

$$H(Y/X) = H(Y) \qquad H(X/Y) = H(X) \qquad I(X \wedge Y) = 0$$

la capacità è sempre nulla e *non è possibile trasmettere informazione*.

4.2.5 Canale simmetrico

È questo il caso più importante e verosimile dal punto di vista applicativo. Se $|\mathcal{X}| = s$ e $|\mathcal{Y}| = m$, la matrice $\boldsymbol{\Gamma}$ è tale che ogni riga è permutazione di ogni altra riga e ogni colonna è permutazione di ogni altra colonna. In tal caso l'ambiguità $H(Y/X = x_i)$ non dipende da x_i, poiché le righe contengono gli stessi elementi. Per il calcolo della capacità scegliamo allora una riga a caso, p.es. la prima:

$$H(Y/X) = \sum_{i=1}^{s} p(x_i)H(Y/X = x_i) = H(Y/X = x_1)$$

$$C = \max_{P} I(X \wedge Y) = \max_{P} [H(Y) - H(Y/X = x_1)]$$

Poiché $H(Y/X = x_i)$ dipende solo dal canale, per rendere massima la mutua informazione si deve scegliere P in modo che la d.p. di $\mathcal{Y}$ sia uniforme. Ciò succede quando anche la d.p. d'ingresso è uniforme:

$$p(y_j) = \sum_{i=1}^{s} p(x_i)p(y_j/x_i) = 1/s \sum_{i=1}^{s} p(y_j/x_i) \qquad (1 \leq j \leq m)$$

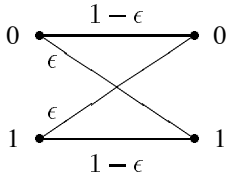
infatti l'ultima sommatoria rappresenta una costante, essendo somma di tutti gli elementi della colonna j-esima. La capacità diventa allora

$$C = \log m - H(Y/X = x_1) \tag{4.12}$$

Esempio 4.2. Consideriamo il caso del più semplice modello possibile, il *canale simmetrico binario* (*CSB*) di figura 4.3. La matrice di transizione è la seguente con $0 \leq \epsilon \leq 1$). La capacità vale

$$C = 1 - H(\epsilon, 1-\epsilon) = 1 + \epsilon \log \epsilon + (1-\epsilon) \log(1-\epsilon) \tag{4.13}$$

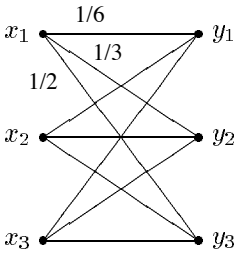
Per $\epsilon = 0$ e $\epsilon = 1$ il canale è senza rumore; per $\epsilon = 1/2$ è inutile, poiché la capacità si annulla. ◯



$$\boldsymbol{\Gamma} = \begin{pmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{pmatrix}$$

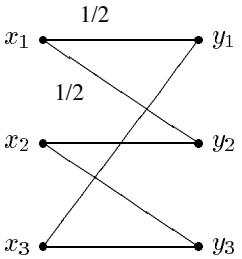
Figura 4.3 Canale simmetrico binario.

Nel seguito sono riportati altri due esempi minimali di canali simmetrici con alfabeto di tre elementi.



$$\begin{pmatrix} 1/6 & 1/3 & 1/2 \\ 1/2 & 1/6 & 1/3 \\ 1/3 & 1/2 & 1/6 \end{pmatrix}$$

$$C = \log 3 - H(\tfrac{1}{2}, \tfrac{1}{3}, \tfrac{1}{6}) = \tfrac{1}{2}\log 3 - \tfrac{2}{3}$$



$$\begin{pmatrix} 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \end{pmatrix}$$

$$C = \log 3 - H(\tfrac{1}{2}, \tfrac{1}{2}) = \log 3 - 1$$

Esempio 4.3. Vediamo ora il *canale simmetrico q-ario*, dove $s = m = q$. La matrice è

$$\begin{pmatrix} 1-\epsilon & \frac{\epsilon}{q-1} & \frac{\epsilon}{q-1} & \cdots & \frac{\epsilon}{q-1} \\ \frac{\epsilon}{q-1} & 1-\epsilon & \frac{\epsilon}{q-1} & & \vdots \\ \vdots & & & & \\ \frac{\epsilon}{q-1} & \cdots & & \cdots & 1-\epsilon \end{pmatrix} \tag{4.14}$$

e la capacità risulta pari a

$$C = \log q + (1-\epsilon)\log(1-\epsilon) + \epsilon \log \frac{\epsilon}{q-1} \tag{4.15}$$

Esprimendo i logaritmi in base q possiamo scrivere

$$C = 1 - H_q(\epsilon) \tag{4.16}$$

dove

$$\begin{aligned} H_q(\epsilon) &= -\epsilon \log_q \epsilon - (1-\epsilon)\log_q(1-\epsilon) + \epsilon \log_q(q-1) \\ &= h_q(\epsilon) + \epsilon \log_q(q-1) \end{aligned} \tag{4.17}$$

(vedi anche (2.30)) è la cosiddetta *Entropia q-aria*. Nella figura 4.4 è rappresentato l'andamento della funzione che ha un minimo (zero) in corrispondenza dell'ascissa $\epsilon = \frac{q-1}{q}$. Solitamente si suppone che sia sempre $\epsilon \leq \frac{q-1}{q}$. Quando $q = 2$ ci si riduce al caso precedente e la curva diventa simmetrica rispetto a $\epsilon = 1/2$.

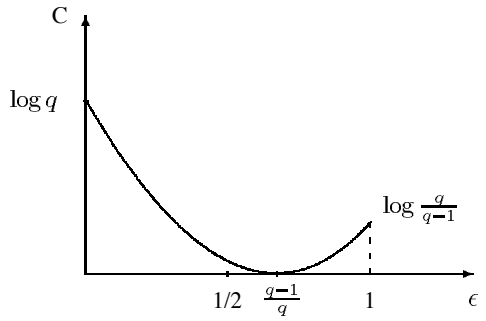
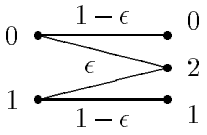


Figura 4.4 Diagramma della capacità di un canale simmetrico q-ario.

4.2.6 Canale simmetrico con cancellazione

Il canale in questione trasforma i bit errati in un terzo simbolo, p.es. un '2', che consente di riconoscere la presenza di un errore. La matrice di transizione assume la forma seguente:

$$\begin{pmatrix} 1-\epsilon & 0 & \epsilon \\ 0 & 1-\epsilon & \epsilon \end{pmatrix}$$



con $\mathcal{X} = \{0,1\}, p(0) = 1-p, p(1) = p, \mathcal{Y} = \{0,1,2\}$.
Poiché le righe sono permutazione l'una dell'altra si ha

$$H(Y/X) = \sum_i \Pr\{X = x_i\} H(Y/X = x_i) = H(Y/X = x_1) = h(\epsilon)$$

e la capacità vale di nuovo

$$C = \max_P I(X \wedge Y) = \max_P H(Y) - H(Y/X = x_1)$$

Ma

$$\begin{array}{l} p(y=0) = \sum_x p(x, y=0) = (1-p)(1-\epsilon) \\ p(y=1) = \sum_x p(x, y=1) = p(1-\epsilon) \\ p(y=2) = \sum_x p(x, y=2) = p\epsilon + (1-p)\epsilon = \epsilon \end{array}$$

e facendo qualche passaggio si ottiene, per l'entropia $H(Y)$

$$H(Y) = h(\epsilon) + (1-\epsilon)h(p) \leq h(\epsilon) + (1-\epsilon)$$

e la limitazione superiore viene raggiunta per una d.p. d'ingresso uniforme $p = 1/2$. La capacità vale dunque

$$C = 1 - \epsilon$$

poiché $H(Y/X = x_1) = h(\epsilon)$.

4.3 Criteri di decodifica

Prima di analizzare la struttura dei codici che consentono la rivelazione e la correzione degli errori, è opportuno descrivere nel suo insieme il problema della codifica e della decodifica di canale, per specificare nel dettaglio il modello matematico che viene impiegato. Ciò porterà allo sviluppo dei teoremi asintotici di codifica di canale, nella loro parte *diretta* e *inversa*; nella parte diretta si dimostra *l'esistenza* di schemi di codifica che consentono di ottenere una probabilità d'errore asintoticamente nulla, purché si mantenga il tasso di trasmissione al di sotto della capacità del canale. Nella parte inversa si verifica che, trasmettendo a un tasso superiore alla capacità, la probabilità d'errore non può essere resa piccola a piacere, neanche con un procedimento asintotico; anzi, nella versione *forte* del teorema inverso si dimostra che la probabilità d'errore tende a 1.

Supponiamo di trovarci all'uscita del codificatore di sorgente; in questo punto del sistema di trasmissione unidirezionale abbiamo a disposizione delle sequenze che sono state sfrondate dalla ridondanza, e che sono costruite sull'alfabeto in uso al canale. La distribuzione di probabilità della v.a. che descrive tale processo è presumibilmente molto vicina a quella uniforme (si veda l'osservazione 3.9), sempre che la compressione dei dati si sia spinta in prossimità dell'entropia. Per fissare le idee supponiamo che la sequenza sia stata codificata su un insieme di M parole di codice q-arie, a lunghezza costante n, in modo tale che ad ogni parola del codice di sorgente venga associata una parola del codice di canale. Il punto

di partenza sta nel fatto che se $M = q^n$, cioè usiamo tutte le possibili n-ple come parole di codice, non c'è alcuna possibilità di accorgersi di un eventuale errore occorso sul canale. In tal caso, infatti, l'effetto dell'errore è quello di trasformare la parola di codice trasmessa in un'altra parola di codice, che verrà poi interpretata come la "vera" parola immessa nel canale. Dovrà essere dunque $M < q^n$, il che corrisponde all'introduzione di una certa *ridondanza*, misurata dall'eccesso d'informazione usata rispetto a quella strettamente necessaria ($\log_q M$) per distinguere in base q M oggetti distinti. La normalizzazione a n di quest'ultima quantità definisce il tasso (4.4) del codice $R = \frac{\log_q M}{n}$ con $R < 1$. Se le parole di codice sono equiprobabili, l'autoinformazione normalizzata a n associata a ogni parola vale proprio $(-\log_q(1/M))/n = R$, assumendo R il significato di quantità *d'informazione in q-it per simbolo di canale*. L'aver introdotto ridondanza consente di tentare il recupero dell'informazione degradata dal rumore. Se $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ è l'insieme delle parole di codice in ingresso al canale, all'uscita dello stesso potremo trovare, per effetto del rumore, una qualunque tra le q^n n-ple possibili appartenenti all'insieme $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{q^M}\}$. Ricevuto dunque $\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}$ dobbiamo elaborare una stima $\widehat{\boldsymbol{x}}$ della parola $\boldsymbol{x}$ trasmessa. Tale operazione, piuttosto delicata, viene fatta dal *decodificatore di canale DC*, che introduce una *partizione* in M classi di equivalenza $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_M$ su $\boldsymbol{\mathcal{Y}}$, in modo da associare ciascun vettore ricevuto a un qualche elemento di $\boldsymbol{\mathcal{X}}$ secondo la *funzione di decodifica* Ψ

$$\Psi : \boldsymbol{\mathcal{Y}} \to \boldsymbol{\mathcal{X}}$$

Le classi di equivalenza $\mathcal{R}_j$ così introdotte si chiamano *regioni di decodifica*, e determinano implicitamente anche la probabilità d'errore dello schema di decodifica (si veda la figura 4.5). Risulta evidente che viene commesso un errore

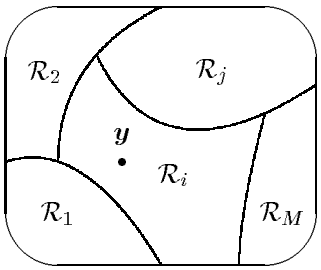


Figura 4.5 Regioni di decodifica.

ogniqualvolta, trasmesso $\boldsymbol{x}_i$, si riceve $\boldsymbol{y} \notin \mathcal{R}_i$, cioè $\boldsymbol{y} \in \bar{\mathcal{R}}_i$ (complementare di

$\mathcal{R}_i$). La probabilità d'errore sul messaggio $\boldsymbol{x}_i$ è dunque data da

$$P_e(\boldsymbol{x}_i) = \sum_{\boldsymbol{y} \in \overline{\mathcal{R}}_i} p(\boldsymbol{y}/\boldsymbol{x}_i)$$

mentre il valore medio, su tutti i messaggi, vale

$$P_e = \sum_{i=1}^{M} p(\boldsymbol{x}_i) P_e(\boldsymbol{x}_i) = \sum_{i=1}^{M} p(\boldsymbol{x}_i) \sum_{\boldsymbol{y} \in \overline{\mathcal{R}}_i} p(\boldsymbol{y}/\boldsymbol{x}_i) \tag{4.18}$$

Poiché le $p(\boldsymbol{x}_i)$ sono di norma assegnate, le $P_e(\boldsymbol{x}_i)$ dipendono dallo schema di decodifica assegnato, e in particolare, oltre che dalla matrice di transizione $\boldsymbol{\Gamma}$, anche dalla disposizione delle regioni $\mathcal{R}_i$. Fare una buona decodifica significa allora scegliere la partizione che consenta di minimare la probabilità media d'errore. Analizzeremo nel seguito le possibili strategie da usare in questo contesto.

4.3.1 Criterio dell'osservatore ideale

Per costruire le regioni di decodifica in modo ottimale dobbiamo partire dal vettore ricevuto $\boldsymbol{y}$, cercando di minimare la probabilità d'errore $P_e(\boldsymbol{y})$. Poiché si verifica un errore quando $\widehat{\boldsymbol{x}} \neq \boldsymbol{x}$, possiamo scrivere

$$P_e(\boldsymbol{y}) = \Pr\{\widehat{\boldsymbol{x}} \neq \boldsymbol{x}/\boldsymbol{y}\} = 1 - \Pr\{\widehat{\boldsymbol{x}} = \boldsymbol{x}/\boldsymbol{y}\}$$

Mediando su tutte le possibili uscite si ottiene

$$P_e = \sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}} p(\boldsymbol{y}) P_e(\boldsymbol{y}) = 1 - \sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}} p(\boldsymbol{y}) \Pr\{\widehat{\boldsymbol{x}} = \boldsymbol{x}/\boldsymbol{y}\} \tag{4.19}$$

La probabilità d'errore può dunque essere minimata scegliendo uno schema di decodifica che renda *massima* la sommatoria sulla destra, che corrisponde alla *probabilità di corretta decodifica associata a* $\boldsymbol{y}$

$$P_c = \Pr\{\widehat{\boldsymbol{x}} = \boldsymbol{x}\} = \sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}} \Pr\{\widehat{\boldsymbol{x}} = \boldsymbol{x}, \boldsymbol{y}\} = \sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}} \Pr\{\widehat{\boldsymbol{x}} = \boldsymbol{x}/\boldsymbol{y}\} p(\boldsymbol{y}) \tag{4.20}$$

dove $p(\boldsymbol{y}) = \sum_{i=1}^{M} p(\boldsymbol{y}/\boldsymbol{x}_i) p(\boldsymbol{x}_i)$ non dipende da i. Ricevuta $\boldsymbol{y}$ si deve allora scegliere una stima $\widehat{\boldsymbol{x}}$ che consenta di rendere massima la probabilità a posteriori di $\boldsymbol{x}$ trasmesso noto $\boldsymbol{y}$ ricevuto. Tra tutte le parole di codice prenderemo allora quella che ha la massima probabilità a posteriori di aver generato $\boldsymbol{y}$.

Ciò porta al seguente criterio:

$$\text{se} \quad \forall j \neq i \quad \text{risulta} \quad p(\boldsymbol{x}_i/\boldsymbol{y}) \geq p(\boldsymbol{x}_j/\boldsymbol{y}) \tag{4.21}$$

allora quando si riceve $\boldsymbol{y}$ si pone $\widehat{\boldsymbol{x}} = \boldsymbol{x}_i$, cioè si colloca $\boldsymbol{y}$ in $\mathcal{R}_i$.

Il criterio dell'osservatore ideale ha l'inconveniente che il decodificatore va progettato tenendo conto delle probabilità a priori $p(\boldsymbol{x}_i)$. Infatti dalla (4.21) si ottiene, sulla base del teorema di Bayes

$$p(\boldsymbol{x}_i/\boldsymbol{y}) = \frac{p(\boldsymbol{x}_i)p(\boldsymbol{y}/\boldsymbol{x}_i)}{p(\boldsymbol{y})}$$

che equivale a porre $\boldsymbol{y}$ in $\mathcal{R}_i$ se $\forall j \neq i$ risulta

$$p(\boldsymbol{x}_i)p(\boldsymbol{y}/\boldsymbol{x}_i) \geq p(\boldsymbol{x}_j)p(\boldsymbol{y}/\boldsymbol{x}_j) \tag{4.22}$$

restando evidente la dipendenza dalle $p(\boldsymbol{x}_i)$. Se queste cambiano diventa necessario ritarare il decodificatore. Si tenga inoltre conto che per certi tipi di canali potrebbe accadere che venga sempre commesso un errore in presenza di un certo ingresso.

4.3.2 Criterio di massima verosimiglianza

Il criterio dell'osservatore ideale tratta probabilità del tipo $p(\boldsymbol{x}_j/\boldsymbol{y})$, che non sono in genere disponibili direttamente a partire dalla matrice di transizione del canale. L'impiego diretto delle probabilità di transizione $p(\boldsymbol{y}/\boldsymbol{x}_j)$ diventa lecito nella (4.22) solo nel caso in cui sia $p(\boldsymbol{x}_i) = 1/M$, $\forall i$, cioè la d.p. d'ingresso sia uniforme. D'altra parte ricordiamo che un buon codificatore di sorgente si comporta come una sorgente D-aria con una d.p. sostanzialmente uniforme, e questa d.p. è proprio quella d'ingresso al canale. Possiamo allora supporre, con una lieve approssimazione tollerabile dal punto di vista applicativo, che sia $p(\boldsymbol{x}_i) = 1/M$, il che porta a una semplificazione del criterio dell'osservatore ideale, che si trasforma in quello di *massima verosimiglianza*:

$$\text{se} \quad \forall j \neq i \quad \text{risulta} \quad p(\boldsymbol{y}/\boldsymbol{x}_i) \geq p(\boldsymbol{y}/\boldsymbol{x}_j) \tag{4.23}$$

allora, quando si riceve $\boldsymbol{y}$ si pone $\widehat{\boldsymbol{x}} = \boldsymbol{x}_i$, cioè si colloca $\boldsymbol{y}$ in $\mathcal{R}_i$. Con questo criterio il problema della ritaratura del decodificatore non sussiste più e si può lavorare direttamente sulla matrice di transizione $\boldsymbol{\Gamma}$. Naturalmente se la d.p. d'ingresso si discosta dall'uniformità, il decodificatore a massima verosimiglianza non è più ottimale.

4.3.3 Criterio dell'errore massimale

Un altro metodo per sbarazzarsi della fastidiosa dipendenza dalla d.p. d'ingresso è quello, più esigente, d'imporre un errore *massimale* δ per tutte le parole di codice

$$P_e(\boldsymbol{x}_i) \leq \delta$$

che conduce a una probabilità media d'errore limitata dallo stesso parametro

$$P_e = \sum_{i=1}^{M} P_e(\boldsymbol{x}_i) p(\boldsymbol{x}_i) \leq \delta$$

Anche se in generale la costruzione effettiva di schemi di decodifica siffatti non risulta agevole, il problema è stato sviluppato e risolto nella sua parte asintotica dal teorema diretto di Shannon; in quest'ambito si dimostra infatti l'esistenza di schemi di decodifica ad errore massimale (e infinitesimo) che consentono di spingere il tasso di trasmissione fino la limite della capacità di canale. La capacità risulta avere così un ruolo centrale nella codifica di canale, analogo per importanza a quello dell'entropia per la codifica di sorgente.

4.4 Teoremi diretto e inverso per la codifica di canale

I teoremi di Shannon costituiscono l'ossatura normativa di tutta la teoria della codifica di canale, stabilendo un risultato, profondo e importante, che riguarda l'esistenza di (successioni di) codici che consentono una probabilità d'errore massimale infinitesima, e ciò purché si usi un tasso di trasmissione *inferiore* alla capacità del canale. Tale risultato non è evidente a priori, ed è stato un grosso merito di Shannon averlo intuito nei suoi tratti essenziali, introducendo l'ingegnoso espediente della *codifica aleatoria*. anche se le prime dimostrazioni rigorose risalgono a qualche anno più tardi. In tal modo si viene a stabilire che il rumore non è limitativo della *precisione* con la quale la trasmissione viene effettuata, ma solamente della sua *velocità*, immediatamente riconducibile al tasso della trasmissione stessa.

Il teorema di codifica di canale si divide in realtà in due parti, una diretta e una inversa (quest'ultima dovuta a Fano). La prima è un risultato di tipo *positivo*, nel senso che attesta l'esistenza di codici che garantiscono una certa prestazione, cioè una probabilità d'errore asintoticamente nulla quando il tasso si mantiene al di sotto del valore della capacità. La seconda fornisce invece un risultato di tipo *negativo*, negando l'esistenza di codici che forniscano tali prestazioni se il tasso supera la capacità del canale; della parte inversa esiste anzi una versione più forte, nella quale si stabilisce che trasmettendo con un tasso superiore alla capacità la probabilità d'errore tende a 1 [36]. Come già osservato precedentemente la capacità diventa la protagonista principale della teoria della codifica di canale, poiché descrive un preciso limite fisico del canale, correlato con la probabilità d'errore ϵ dello stesso.

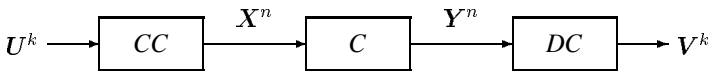


Figura 4.6 Le variabili implicate nel teorema inverso.

4.4.1 Parte inversa

Iniziamo con l'illustrare il risultato negativo, che esprime in modo esplicito ciò che *non* è possibile attendersi da un canale di capacità C determinata. Facciamo riferimento allo schema di figura 4.6. Sia $\boldsymbol{u}^k = (u_1, u_2, \ldots, u_k)$ una sequenza di k lettere uscenti dalla sorgente, dove indichiamo con $\boldsymbol{U}^k = (U_1, U_2, \ldots, U_k)$ il vettore di v.a. che descrive il processo. Esso viene successivamente elaborato dal *codificatore di canale CC*, che aggiunge $n - k$ bit di controllo. L'uscita del codificatore si può rappresentare per mezzo della sequenza $\boldsymbol{x}^n = (x_1, x_2, \ldots, x_n)$, associata al vettore di v.a. $\boldsymbol{X}^n = (X_1, X_2, \ldots, X_n)$, che viene immessa nel canale stazionario e senza memoria con matrice di transizione $\boldsymbol{\Gamma}$. All'uscita del canale si trova la sequenza $\boldsymbol{y}^n = (y_1, y_2, \ldots, y_n)$, associata al vettore di v.a. $\boldsymbol{Y}^n = (Y_1, Y_2, \ldots, Y_n)$, che viene accettato all'ingresso del decodificatore di canale; quest'ultimo ha il compito di fornire una stima $\widehat{\boldsymbol{u}}^k = \boldsymbol{v}^k = (v_1, v_2, \ldots, v_k)$, associata al vettore di v.a. $\boldsymbol{V}^k = (V_1, V_2, \ldots, V_k)$, di quello che è stato trasmesso.

Il codice di canale impiegato nel sistema di comunicazione è caratterizzato dal *dizionario* $\mathfrak{C} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ di M parole, e da una *regola di decodifica* che pone ciascun vettore $\boldsymbol{y}$ nella regione di decodifica appropriata $\mathcal{R}_i$. Prima di affrontare il teorema vero e proprio forniremo un lemma funzionale alla dimostrazione.

Lemma 4.1. *In un canale finito, stazionario e senza memoria avente capacità C risulta*

$$I(\boldsymbol{X}^n \wedge \boldsymbol{Y}^n) \leq nC \tag{4.24}$$

Dim. Per la stazionarietà e l'assenza di memoria vale la (2.28), che assieme alla definizione (4.10) di capacità porta a

$$I(\boldsymbol{X}^n \wedge \boldsymbol{Y}^n) \leq \sum_{i=1}^{n} I(X_i \wedge Y_i) \leq \sum_{i=1}^{n} \max I(X_i \wedge Y_i) \leq nC$$

Si noti che, sempre per effetto del teorema (2.5), la prima disuguaglianza sussiste come uguaglianza quando le v.a. $Y_1, Y_2, \ldots, Y_n$ sono indipendenti. □

Consideriamo ora l'entropia media per lettera di sorgente, che risulta essere pari a

$$H_k(U) = \frac{H(\boldsymbol{U^k})}{k} = -\frac{1}{k}\sum_{\boldsymbol{u}} p(\boldsymbol{u^k}) \log p(\boldsymbol{u^k})$$

Si ricordi che per una sorgente stazionaria $H_k(U)$ è non crescente con k quando $k \to \infty$, e $H_k(U) \to H_\infty(U)$. Se la sorgente è anche senza memoria si ha $H_k(U) = H(U)$. È evidente che l'obiettivo del sistema di comunicazione è quello di fare in modo che sia $\boldsymbol{v}^k = (v_1, v_2, \ldots, v_k) = \boldsymbol{u}^k = (u_1, u_2, \ldots, u_k)$. Se per qualche i succede che $u_i \neq v_i$, allora si è verificato un errore nella posizione i-esima; sia $P_{e/i}$ la probabilità corrispondente. Il valore medio della probabilità d'errore risulta essere

$$P_e = \frac{1}{k}\sum_{i=1}^{k} P_{e/i}$$

anche se in pratica il valore $P_{e/i}$ è in buona sostanza indipendente da i. Il numero di errori attesi su una sequenza di lunghezza k è allora kP_e che, per k sufficientemente grande, diventa maggiore di 1. Ciò significa che $P_e \geq 1/k \to 0$ quando $k \to \infty$. Nel teorema inverso si deve dimostrare che se il tasso supera la capacità, la probabilità d'errore si mantiene maggiore di zero in senso stretto, cioè $P_e \geq cost > 0$, e l'approccio appena seguito non è d'aiuto. È necessario lavorare direttamente sulla probabilità media d'errore. Iiniziamo col caso $k = 1$ e i vettori generati dalla sorgente si riducono a delle semplici lettere sull'alfabeto q-ario del canale. Se trasmettiamo u e riceviamo v la probabilità d'errore vale

$$P_{e/i} = \Pr\{U_i \neq V_i\} = \sum_{u}\sum_{v \neq u} p(u, v)$$

Poiché le variabili aleatorie U_i e V_i assumono i rispettivi valori sullo stesso alfabeto, ed è nota la probabilità che si realizzi un errore, possiamo usare la disuguaglianza di Fano (2.30) per limitare l'incertezza media su U nota che sia la v.a. V, che possiamo considerare come stima di U. Ciò porta alla relazione

$$H(U_i/V_i) \leq h(P_{e/i}) + P_{e/i} \log(q-1) \tag{4.25}$$

Questa limitazione può essere interpretata secondo quanto riportato dalla figura 2.2, nella quale compare il diagramma della funzione $h(P_{e/i}) + P_{e/i} \log(q-1)$, dove $\epsilon = P_{e/i}$ e $K = q$. Come si può vedere, imponendo per l'equivocazione un valore $H^*(U/V)$, la disuguaglianza può essere soddisfatta solo quando

la probabilità d'errore si mantenga al di sopra dell'ascissa corrispondente $P^*_{e/i}$. Passiamo ora all'analisi del caso con $k \geq 1$, che intuitivamente dovrebbe portare a un risultato analogo per le variabili aleatorie vettoriali $\boldsymbol{U}^k = (U_1, U_2, \ldots, U_k)$ e $\boldsymbol{V}^k = (V_1, V_2, \ldots, V_k)$.

Consideriamo allora l'equivocazione $H(\boldsymbol{U}^k/\boldsymbol{V}^k)$ e sviluppiamola tenendo conto delle solite regole sul condizionamento:

$$H(\boldsymbol{U}^k/\boldsymbol{V}^k) = H(U_1, U_2, \ldots, U_k/\boldsymbol{V}^k) = H(U_1/\boldsymbol{V}^k) + H(U_2/U_1\boldsymbol{V}^k) + \ldots$$
$$\ldots + H(U_k/U_1U_2\ldots U_{k-1}\boldsymbol{V}^k) \leq \sum_{i=1}^{k} H(U_i/V_i) \leq$$
$$\leq \sum_{i=1}^{k} \left[h(P_{e/i}) + P_{e/i}\log(q-1)\right] \leq k\left[h(P_e) + P_e\log(q-1)\right]$$

dove la penultima disuguaglianza segue dall'applicazione della (4.25). Confrontando gli estremi della catena e dividendo per k si ottiene

$$\frac{1}{k}H(\boldsymbol{U}^k/\boldsymbol{V}^k) \leq h(P_e) + P_e\log(q-1) \tag{4.26}$$

L'ultima disuguaglianza della catena precedente comporta la seguente verifica

$$\frac{1}{k}\sum_{i=1}^{k} h(P_{e/i}) \leq h(P_e) = h\left(\frac{1}{k}\sum_{i=1}^{k} P_{e/i}\right)$$

che può essere fatta agevolmente ricorrendo alla *disuguaglianza della somma logaritmica* (2.2); si ha infatti

$$\frac{1}{k}\sum_{i=1}^{k} h(P_{e/i}) = -\frac{1}{k}\left[\sum_{i=1}^{k} P_{e/i}\log P_{e/i} + (1-P_{e/i})\log(1-P_{e/i})\right] \leq$$
$$\leq -\left[\frac{\sum_{i=1}^{k}(P_{e/i})}{k}\log\frac{\sum_{i=1}^{k}(P_{e/i})}{k} + \frac{\sum_{i=1}^{k}(1-P_{e/i})}{k}\log\frac{\sum_{i=1}^{k}(1-P_{e/i})}{k}\right] =$$
$$= -P_e\log P_e - (1-P_e)\log(1-P_e) = h(P_e)$$

Riprendendo ora la (4.26) si ottiene

$$h(P_e) + P_e\log(q-1) \geq \frac{1}{k}H(\boldsymbol{U}^k/\boldsymbol{V}^k) = \frac{H(\boldsymbol{U}^k)}{k} - \frac{I(\boldsymbol{U}^k \wedge \boldsymbol{V}^k)}{k} \geq$$
$$\geq H_k(U) - \frac{I(\boldsymbol{X}^n \wedge \boldsymbol{Y}^n)}{k} \geq H_k(U) - \frac{n}{k}C \geq H_\infty(U) - \frac{n}{k}C$$

Le ultime tre disuguaglianze derivano rispettivamente dal teorema di elaborazione dati 2.8 ($I(\boldsymbol{U}^k \wedge \boldsymbol{V}^k) \leq I(\boldsymbol{X}^n \wedge \boldsymbol{Y}^n)$ poiché $\boldsymbol{U}^k, \boldsymbol{V}^k$ sono variabili interne rispetto a $\boldsymbol{X}^n, \boldsymbol{Y}^n$), dal fatto che per un canale discreto e senza memoria vale la (4.24) e dalla non crescenza di $H_k(U)$. Le ultime relazioni ci consentono di enunciare il celebre

Teorema 4.1 (Inverso di Fano). *Se* $H_\infty(U) > \frac{n}{k}C$ *la probabilità d'errore rimane limitata inferiormente da una costante maggiore di* 0.

Dim. Dall'ultima catena di disuguaglianze ricaviamo

$$h(P_e) + P_e \log(q-1) \geq H_\infty(U) - \frac{n}{k}C > 0 \tag{4.27}$$

e la P_e non può scendere al di sotto dell'ascissa corrispondente a $H_\infty(U) - \frac{n}{k}C$ (si veda fig. 4.7)□

Il teorema precedente può essere reinterpretato alla luce della definizione

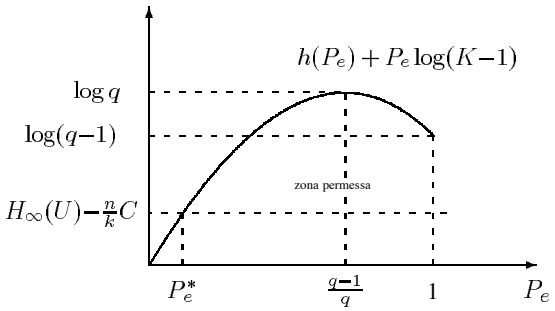


Figura 4.7 Diagramma della funzione $h(P_e) + P_e \log(q-1)$.

di *tasso d'informazione del canale* $R_I = \frac{H(X_1, X_2, \ldots X_n)}{n}$. In tal caso possiamo scrivere

$$h(P_e) + P_e \log(q-1) \geq \frac{n}{k}\left(\frac{H(\boldsymbol{U}^k)}{n} - C\right) =$$
$$= \frac{n}{k}\left(\frac{H(\boldsymbol{X}^n)}{n} - C\right) = \frac{n}{k}(R_I - C)$$

mettendo così in evidenza il legame diretto tra tasso d'informazione del canale e capacità dello stesso. Nell'ipotesi di tasso del canale maggiore della capacità il teorema stabilisce allora, tramite la funzione $h(P_e)+P_e \log(q-1)$ di fig. 4.7, una limitazione inferiore per la probabilità media di errore per lettera di sorgente in una sequenza di k lettere. La probabilità d'errore media non può dunque essere arbitrariamente piccola. Si noti per inciso che, se $R = \frac{\log_q M}{n}$ è il tasso del codice, si ha sempre $R \geq R_I$, come si verifica immediatamente non appena si noti che $R - R_I = \frac{D(P/\!/U)}{n} \geq 0$ (P è la d.p. sulle parole di codice e U è la d.p. uniforme). L'uguaglianza si realizza quando $P \equiv U$. Se $R > C$ il teorema vale a maggior ragione.

4.4.2 Parte diretta

Affronteremo ora il caso in cui il tasso del canale sia inferiore alla capacità, dimostrando l'esistenza di (successioni di) codici che consentono una probabilità d'errore massimale limitata e infinitesima. Appare evidente che in questa circostanza la (4.27) del teorema inverso non soccorre, essendo soddisfatta per tutti i valori di P_e. Prima di dare una delle dimostrazioni disponibili in letteratura, cercheremo di fornire una giustificazione euristica del risultato. Ci riferiremo per semplicità al caso binario, anche se l'estensione al caso più generale è immediata.

Supponiamo di scegliere a caso e di trasmettere sul canale una sequenza $\boldsymbol{x}^n = (x_1, x_2, \ldots, x_n)$, le cui componenti X_i siano indipendenti e associate a una d.p. che raggiunge la capacità. All'uscita troveremo una sequenza $\boldsymbol{y}^n = (y_1, y_2, \ldots, y_n)$. Con questo procedimento aleatorio le sequenze disponibili asintoticamente in ingresso sono "solo" quelle tipiche (cfr. 3.3), che sono in numero di pari a (circa) $2^{nH(X)}$ e con una probabilità uniforme pari a (circa) $2^{-nH(X)}$. Similmente le sequenze di uscita dal canale sono le $2^{nH(Y)}$ tipiche. Tuttavia, dal punto di vista della trasmissione, le sequenze che ci interessano effettivamente sono le $\boldsymbol{x}e\boldsymbol{y}$ *congiuntamente tipiche*, tali cioè che

$$\left|-\frac{1}{n}\log p(\boldsymbol{x}) - H(X)\right| < \delta \qquad \left|-\frac{1}{n}\log p(\boldsymbol{y}) - H(Y)\right| < \delta$$
$$\left|-\frac{1}{n}\log p(\boldsymbol{x},\boldsymbol{y}) - H(X,Y)\right| < \delta$$

Il loro numero è invece pari a $2^{nH(X,Y)}$. Una coppia X, Y congiuntamente tipica viene generata quando si seleziona una Y tipica in uscita che deriva da una X tipica in ingresso tale che X e Y siano congiuntamente tipiche (si veda figura 4.8). Poiché $H(X,Y) = H(X) + H(Y/X) < H(X) + H(Y)$ (X e Y non sono indipendenti!) si ha anche $2^{nH(X,Y)} < 2^{nH(X)}2^{nH(Y)}$, e quindi *non tutte le coppie di sequenze tipiche sono congiuntamente tipiche*. Di conseguenza, per ogni sequenza tipica Y ricevuta ci sono mediamente

$$2^{nH(X,Y)}/2^{nH(Y)} = 2^{nH(X/Y)}$$

sequenze tipiche d'ingresso tali che X e Y siano congiuntamente tipiche. Supponiamo ora di formare un codice scegliendo a caso, tra le $2^{nH(X)}$ tipiche d'ingresso, 2^{nR} $(R < C)$ parole di codice. Se nel canale viene trasmessa la parola di codice $\boldsymbol{x}_i$ e si riceve $\boldsymbol{y}$, un errore di decodifica è possibile quando una o più parole di codice $\boldsymbol{x}_j (j \neq i)$ appartengono alla regione di decodifica $\mathcal{R}_{\boldsymbol{y}}$ che contiene $\boldsymbol{y}$. La corrispondente probabilità d'errore risulta essere pari a

$$\Pr\{\exists \text{ almeno un } \boldsymbol{x}_j, j \neq i : \boldsymbol{x}_j \in \mathcal{R}_{\boldsymbol{y}}\} \leq \sum_{j=1, j\neq i}^{2^{nR}} \Pr\{\boldsymbol{x}_j \in \mathcal{R}_{\boldsymbol{y}}\} \leq$$
$$\leq 2^{nR}\frac{2^{nH(X/Y)}}{2^{nH(X)}} = 2^{n(R-I(X,Y))} = 2^{n(R-C)} \to 0$$

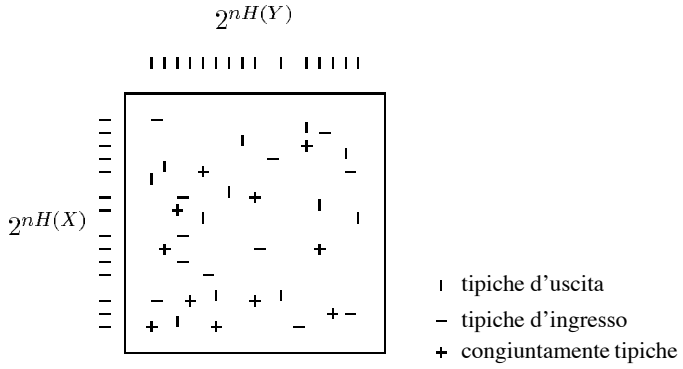


Figura 4.8 Rappresentazione schematica delle sequenze tipiche coinvolte nel teorema diretto.

quando $n \to \infty$, e l'ultima uguaglianza si giustifica dall'ipotesi che la d.p. di X sia tale da raggiungere la capacità. Dunque, nelle ipotesi viste la probabilità d'errore è limitata superiormente da una quantità che tende asintoticamente a zero. Calcoliamo ora il numero di parole di codice compatibili con queste prestazioni. Per ogni sequenza tipica X d'ingresso ci sono circa

$$2^{nH(X,Y)}/2^{nH(X)} = 2^{nH(Y/X)}$$

possibili sequenze tipiche d'uscita con probabilità uniforme. Per poter codificare in modo corretto è necessario che non ci siano due sequenze $\boldsymbol{x}_i, \boldsymbol{x}_j$ che producono la stessa uscita $\boldsymbol{y}$, altrimenti non saremo in grado di decidere qual e $\boldsymbol{x}$ sia stata trasmessa. Poiché il numero totale di sequenze $\boldsymbol{y}$ è $2^{nH(Y)}$ e tale insieme deve essere diviso in sottinsiemi di cardinalità $2^{nH(Y/X)}$ corrispondenti alle diverse sequenze d'ingresso, il numero totale d'insiemi disgiunti sarà al più

$$\frac{2^{nH(Y)}}{2^{nH(Y/X)}} = 2^{nI(X,Y)} = 2^{nC} = \frac{2^{nH(X)}}{2^{nH(X/Y)}}$$

Di conseguenza possiamo associare al più 2^{nC} sequenze alle parole di codice.

Daremo ora una dimostrazione rigorosa del teorema, riferita per semplicità al caso di un canale simmetrico binario, e basata su quella illustrata nel testo di van Lint [84].

Supponiamo di disporre di un codice il cui dizionario contiene M parole $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ a d.p. uniforme e decodifichiamo sulla base del principio di massima verosimiglianza, che come vedremo nella sezione 6.2.2 porta a un criterio di *distanza minima di Hamming*. Sia $P_e(\boldsymbol{x_i})$ la probabilità di compiere un errore di decodifica se viene trasmessa la parola $\boldsymbol{x}_i$. La probabilità media

di errata decodifica si ricava dalla (4.18), tenuto conto dell'uniformità della d.p. d'ingresso, risulta

$$P_e = \frac{1}{M}\sum_{i=1}^{M} P_e(i)$$

Consideriamo ora l'insieme di *tutti i possibili codici* con i parametri assegnati M, n, ϵ per il *CSB(ϵ)* e sia

$$P^*(M, n, \epsilon) = \min P_e$$

Nel seguito dimostreremo che, per ogni $\delta > 0$ e per n sufficientemente grande, esiste un codice $\mathfrak{C}$ con M parole di codice di lunghezza n e con un tasso prossimo alla capacità, tale che $P_e < \delta$. Ciò significa che esistono asintoticamente degli schemi di decodifica che consentono una probabilità d'errore massimale limitata, e ciò a un tasso di trasmissione dell'informazione prossimo alla capacità del canale. Prima di procedere alla dimostrazione vera e propria premettiamo alcuni risultati parziali funzionali alla dimostrazione. Se l' n-pla d'errore ha peso w la sua probabilità, per un canale simmetrico binario, è pari a $\epsilon^w(1-\epsilon)^{n-w}$ e dipende solo da w. Il numero di errori ricevuti è una variabile aleatoria che ha speranza matematica $n\epsilon$ e varianza pari a $n\epsilon(1-\epsilon)$. Se poniamo $\alpha \triangleq \sqrt{\frac{n\epsilon(1-\epsilon)}{\delta/2}}$ per la disuguaglianza di Čebišev si ottiene

$$\Pr\{w > n\epsilon + \alpha\} = \Pr\{w - n\epsilon > \alpha\} \leq \frac{n\epsilon(1-\epsilon)}{\left(\sqrt{\frac{n\epsilon(1-\epsilon)}{\delta/2}}\right)^2} = \frac{\delta}{2} \tag{4.28}$$

Inoltre, posto $\rho = \lfloor n\epsilon + \alpha \rfloor$, indichiamo con

$$S_\rho(\boldsymbol{x}) = \{\boldsymbol{y} : d_H(\boldsymbol{x}, \boldsymbol{y}) \leq \rho\}$$

la *sfera di Hamming* di raggio ρ centrata nella parola $\boldsymbol{x}$. Il suo volume, nel caso binario, è pari a

$$\mathrm{Vol}[S_\rho(\boldsymbol{x})] = |S_\rho(\boldsymbol{x})| = \sum_{i=0}^{\rho} \binom{n}{i} \tag{4.29}$$

(si veda anche la (4.36)) e viene calcolato contando il numero di n-ple che costituiscono i successivi "gusci" a distanza i. Poiché non è agevole trattare con i coefficienti binomiali, possiamo usare la limitazione superiore (3.43) (ridotta al caso $K = 2$)

$$\binom{n}{\rho} \leq 2^{nH(\frac{\rho}{n})}$$

che porta per la (4.29) alla seguente relazione

$$|S_\rho(\boldsymbol{x})| = \sum_{i=0}^{\rho} \binom{n}{i} < \frac{n}{2}\binom{n}{\rho} \leq \frac{n}{2}\, 2^{nH(\frac{\rho}{n})} \tag{4.30}$$

dove la prima disuguaglianza deriva dal fatto che per $\epsilon < 1/2$ ed n sufficientemente grande si ha $\rho < n/2$.

Introduciamo infine le due seguenti funzioni. Siano $\boldsymbol{u}$ e $\boldsymbol{v}$ n-ple e poniamo

$$f(\boldsymbol{u},\boldsymbol{v}) = \begin{cases} 0 & \text{se} \quad d_H(\boldsymbol{u},\boldsymbol{v}) > \rho \\ 1 & \text{se} \quad d_H(\boldsymbol{u},\boldsymbol{v}) \leq \rho \end{cases}$$

Inoltre, se $\boldsymbol{x}_i \in \mathfrak{C}$ e $\boldsymbol{y}$ è una sequenza qualunque definiamo

$$g_i(\boldsymbol{y}) = 1 - f(\boldsymbol{y},\boldsymbol{x}_i) + \sum_{i \neq j} f(\boldsymbol{y},\boldsymbol{x}_j) \tag{4.31}$$

Si noti che $g_i(\boldsymbol{y})$ vale 0 se e solo se $\boldsymbol{x}_i$ è l'unica parola di codice che sta all'interno della sfera ($d_H(\boldsymbol{y},\boldsymbol{x}_i) \leq \rho$). In ogni altro caso si ha $g_i(\boldsymbol{y}) \geq 1$. In altre parole questa funzione segnala, con un valore ≥ 1, la presenza di *un errore di decodifica*, che può derivare tanto dal fatto di non essere in grado di scegliere la parola di codice esatta nel caso ce ne fosse più d'una all'interno della sfera, quanto dal trovarsi senza parole di codice all'interno della stessa sfera. Ricordiamo che la capacità di un canale simmetrico binario è data dalla (4.13), cioè

$$C = 1 - H(\epsilon) = 1 + \epsilon \log \epsilon + (1-\epsilon)\log(1-\epsilon) \tag{4.32}$$

Teorema 4.2 (Diretto di Shannon). *Se* $0 < R < C = 1 - H(\epsilon)$ *e* $M_n = 2^{nR}$, *allora* $P^*(M,n,\epsilon) \to 0$ *quando* $n \to \infty$.

Dim. Dimostreremo che per $\delta > 0$ e n sufficientemente grande, esiste un codice $\mathfrak{C}$ per parole di lunghezza n con un tasso prossimo alla capacità e tale che $P_e < \delta$. Prendiamo M parole di codice $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ scegliendole a caso nell'insieme $\mathbf{2}^n$ di tutte e sole le 2^n sequenze possibili; la decodifica avviene come segue: se riceviamo $\boldsymbol{y}$ e c'è esattamente una parola di codice $\boldsymbol{x}_i$ tale che $d_H(\boldsymbol{x}_i,\boldsymbol{y}) \leq \rho$ allora decodifichiamo $\boldsymbol{y}$ come $\boldsymbol{x}_i$; altrimenti dichiariamo errore (oppure, dovendo per forza decodificare, decodifichiamo sempre p.es. come $\boldsymbol{x}_1$). Se ora $P_e(i)$ è al solito la probabilità di compiere un errore quando viene trasmessa la parola $\boldsymbol{x}_i$ possiamo limitarla nel modo seguente

$$\begin{aligned} P_e(i) &\leq \sum_{\boldsymbol{y}\in\mathbf{2}^n} \Pr\{\boldsymbol{y}/\boldsymbol{x}_i\} g_i(\boldsymbol{y}) = \\ &= \sum_{\boldsymbol{y}\in\mathbf{2}^n} \Pr\{\boldsymbol{y}/\boldsymbol{x}_i\}[1 - f(\boldsymbol{y},\boldsymbol{x}_i)] + \sum_{\boldsymbol{y}\in\mathbf{2}^n} \sum_{j=1, j\neq i}^{M} \Pr\{\boldsymbol{y}/\boldsymbol{x}_i\} f(\boldsymbol{y},\boldsymbol{x}_j) \end{aligned}$$

Poiché $f(\boldsymbol{y}, \boldsymbol{x}_i) = 0$ se $d_H(\boldsymbol{x}_i, \boldsymbol{y}) > \rho$ il primo membro della somma rappresenta la probabilità che la sequenza ricevuta $\boldsymbol{y}$ non stia nella sfera $S_\rho(\boldsymbol{x_i})$ di raggio ρ centrata in $\boldsymbol{x}_i$, e questa probabilità è al più pari a $\delta/2$ per quanto visto sopra nella (4.28). Facendo la media su tutte le parole di codice si ha dunque

$$P_e \leq \frac{\delta}{2} + \frac{1}{M} \sum_{i=1}^{M} \sum_{\boldsymbol{y} \in \boldsymbol{2}^n} \sum_{j \neq i} \Pr\{\boldsymbol{y}/\boldsymbol{x}_i\} f(\boldsymbol{y}, \boldsymbol{x}_j)$$

Consideriamo ora tutti i possibili codici, e per ciascuno di essi ripetiamo il ragionamento fatto, calcolando la speranza matematica $E[P_e]$ della probabilità media d'errore. Per la definizione di $P^*(M, n, \epsilon)$ si ha però

$$P^*(M, n, \epsilon) \leq E[P_e] \leq \frac{\delta}{2} + \frac{1}{M} \sum_{\boldsymbol{y} \in \boldsymbol{2}^n} \sum_{i=1}^{M} \sum_{j \neq i} E[\Pr\{\boldsymbol{y}/\boldsymbol{x}_i\}] E[f(\boldsymbol{y}, \boldsymbol{x}_j)] \tag{4.33}$$

e, per la scelta casuale delle parole di codice sulla base di una distribuzione uniforme, possiamo scrivere

$$E[f(\boldsymbol{y}, \boldsymbol{x}_j)] = \sum_{\boldsymbol{z} \in \boldsymbol{2}^n} f(\boldsymbol{y}, \boldsymbol{z}) \Pr\{\boldsymbol{x}_j = \boldsymbol{z}\} = \frac{1}{2^n} \sum_{\boldsymbol{z} \in \boldsymbol{2}^n} f(\boldsymbol{y}, \boldsymbol{z}) = \frac{|S_\rho(\boldsymbol{y})|}{2^n}$$

per la seconda speranza matematica e

$$E[\Pr\{\boldsymbol{y}/\boldsymbol{x}_i\}] = \sum_{\boldsymbol{z} \in \boldsymbol{2}^n} \Pr\{\boldsymbol{y}/\boldsymbol{z}\} \Pr\{\boldsymbol{x}_i = \boldsymbol{z}\} = \frac{1}{2^n} \sum_{\boldsymbol{z} \in \boldsymbol{2}^n} \Pr\{\boldsymbol{y}/\boldsymbol{z}\}$$

per la prima. Sostituendo questi valori nella (4.33) si ottiene

$$P^*(M, n, \epsilon) \leq \frac{\delta}{2} + \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} \sum_{\boldsymbol{y} \in \boldsymbol{2}^n} \frac{|S_\rho(\boldsymbol{y})|}{2^n} \left[\frac{1}{2^n} \sum_{\boldsymbol{z} \in \boldsymbol{2}^n} \Pr\{\boldsymbol{y}/\boldsymbol{z}\} \right] \tag{4.34}$$

Si noti che

$$\sum_{\boldsymbol{z} \in \boldsymbol{2}^n} \sum_{\boldsymbol{y} \in \boldsymbol{2}^n} \Pr\{\boldsymbol{y}/\boldsymbol{z}\} = \sum_{\boldsymbol{z} \in \boldsymbol{2}^n} 1 = 2^n$$

e dunque

$$\sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} \sum_{\boldsymbol{y} \in \boldsymbol{2}^n} \sum_{\boldsymbol{z} \in \boldsymbol{2}^n} \Pr\{\boldsymbol{y}/\boldsymbol{z}\} = \sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} 2^n = M(M-1)2^n$$

Sostituendo nella (4.34) e semplificando si ricava

$$P^*(M, n, \epsilon) - \frac{\delta}{2} \leq \frac{M-1}{2^n} |S_\rho| \tag{4.35}$$

poiché $|S_\rho(\boldsymbol{y})| = |S_\rho(\mathbf{0})| \triangleq |S_\rho| \ \forall \boldsymbol{y} \in \mathbf{2}^n$. Prendendo il logaritmo, dividendo per n e ricordando la (4.30) si ottiene

$$\frac{1}{n}\log\left[P^*(M,n,\epsilon) - \frac{\delta}{2}\right] \leq \frac{1}{n}\log(M-1) - 1 + \frac{1}{n}\log|S_\rho| < \\ < \frac{1}{n}\log M - 1 + H(\frac{\rho}{n}) + \frac{1}{n}\log\frac{n}{2}$$

Ora, quando $n \to \infty$ si ha

$$\frac{\rho}{n} = \frac{\lfloor n\epsilon + \alpha \rfloor}{n} \to \epsilon$$

e dunque, ricordando la (4.32), si ha $1 - H(\frac{\rho}{n}) \to 1 - H(\epsilon) = C$, capacità del canale simmetrico binario. Sostituendo $M = M_n$ e ponendo $\zeta_n = \frac{1}{n}\log\frac{n}{2}$ infinitesimo, possiamo scrivere

$$\frac{1}{n}\log\left[P^*(M,n,\epsilon) - \frac{\delta}{2}\right] \leq R - C + \zeta_n < -\beta < 0$$

essendo $R < C$ strettamente. Ciò consente di limitare la probabilità d'errore, per $n > n_0$ opportuno

$$P^*(M,n,\epsilon) < \frac{\delta}{2} + 2^{-\beta n}$$

che prova il teorema. □

Oss. 4.1. La dimostrazione appena vista per il caso binario si estende facilmente al caso q-ario non appena si consideri, oltre alla capacità (4.16) del canale q-ario, una limitazione superiore idonea per il volume

$$\text{Vol}[S_\rho(\boldsymbol{x})] = |S_\rho(\boldsymbol{x})| = \sum_{i=0}^{\rho}\binom{n}{i}(q-1)^i \tag{4.36}$$

della sfera q-aria di Hamming. Tale limitazione sarà calcolata nel paragrafo 6.6.3 e vale (rapportata al tasso)

$$\frac{1}{n}\log_q|S_\rho(\boldsymbol{x})| \leq H_q(\frac{\rho}{n}) + \frac{1}{n}\log_q(n+1) \tag{4.37}$$

dove $H_q(\alpha)$ è *l'entropia q-aria* già incontrata nella (4.17).

Oss. 4.2. La dimostrazione del teorema non ha carattere costruttivo. Come vedremo ampiamente nella parte dedicata ai codici correttori il fatto che scegliendo a caso le parole di codice si ottenga un "buon" codice aiuta poco o nulla sul piano attuativo, poiché nella pratica si ha bisogno di codici dotati di una qualche *struttura*, che consenta di effettuare in modo efficiente le operazioni di codifica/decodifica evitando la consultazione diretta del dizionario.

Parte II

Codici

Capitolo 5
Codici di sorgente

5.1 Introduzione

Nella prima parte ci siamo occupati del problema della codifica di sorgente e di canale da un punto di vista normativo. La codifica di sorgente consiste nella traduzione dell'informazione dall'alfabeto K-ario di sorgente a quello D-ario compatibile col funzionamento del canale, traduzione effettuata secondo opportuni criteri di economicità. È infatti necessario rendere minime le lunghezza medie delle sequenze che escono dal codificatore di sorgente al fine di minimare il tempo di occupazione del canale, ovvero lo spazio di memoria destinato alla registrazione dei dati generati. L'obiettivo viene raggiunto sfrondando la ridondanza (sintattica) relativa alle sequenze primarie e compattando tutta l'informazione su un nucleo non ulteriormente comprimibile.

Per quanto riguarda la codifica di canale l'obiettivo è invece quello di inserire ridondanza strutturale all'interno dei messaggi che dovranno essere trasmessi sul canale, col fine di aumentare la resistenza di questi ultimi al rumore, che inevitabilmente colpisce qualunque mezzo fisico di trasmissione dati.

In entrambi i casi i risultati ottenuti hanno quasi sempre valore normativo, stabilendo il comportamento (spesso asintotico) dei parametri che descrivono le prestazioni di detti codici. I teoremi sono di tipo *diretto*, nel qual caso assicurano l'esistenza di codici con prestazioni minime garantite, o di tipo *inverso*, se stabiliscono la non esistenza di codici con parametri migliori di quelli garantiti. Quasi mai la descrizione data in questo contesto è di tipo costruttivo. Rimane dunque aperto il problema della *costruzione effettiva* dei codici promessi dai vari teoremi diretti di codifica, tanto per la codifica di sorgente che per quella di canale.

Nel caso della codifica di sorgente si è visto che, a prescindere dalla modalità di effettuazione (*B-LV, B-B, LV-B, LV-LV*), la compressione dei dati non può essere spinta oltre l'entropia della sorgente, pena la perdita dell'univoca decodificabilità del codice. Per il caso particolare dei codici *B-B* l'entropia può essere raggiunta solo rinunciando alla condizione u.d. e a spese di una

probabilità d'errore strettamente maggiore di 0 (anche se infinitesima asintoticamente).

si ha addirittura che la sola richiesta di raggiungimento dell'entropia comporta la perdita dell'univoca decodificabilità, anche se l'errore introdotto può essere limitato asintoticamente da un infinitesimo.

Nel presente capitolo ci occuperemo esplicitamente dell'individuazione dei codici *ottimi* per la codifica *B-LV* e *LV-B*, cioè dei codici che consentono di effettuare, a parità di condizioni, la massima compressione dei dati. Questa ricerca verrà effettuata nell'ipotesi artificiosa di stazionarietà e assenza di memoria per la sorgente (sorgente *SSM*). L'utilità di quest'impostazione così rigida deriva dalla circostanza che, una volta noti i codici ottimi, si possono allentare certi vincoli per individuare codici che vanno abbastanza bene anche nel caso più generale. Un esempio di ciò lo vedremo nel paragrafo 5.5, nel quale la struttura del *codice ottimo B-LV di Huffman* verrà utilmente impiegata al di fuori delle ipotesi restrittive (*SSM*) grazie alle quali essa è stata concepita.

In tutti questi casi si presuppone comunque di avere una *descrizione statistica* della sorgente e di sfruttarla pienamente per effettuare la compressione dei dati. Vedremo però che è possibile concepire degli schemi di codifica assolutamente svincolati dalla statistica della sorgente (tanto nel caso *B-LV* che in quello *LV-B*), e che consentono una compressione *algoritmica* o *strutturale* delle sequenze. Questi sono i cosiddetti *codici universali*, di cui faremo menzione nel seguito.

Per quanto riguarda invece la codifica di canale, descriveremo la struttura (algebrica) dei principali *codici correttori d'errore*, che consentono di correggere alcune configurazioni d'errore o quantomeno di rilevarne la presenza. Una strutturazione soggiacente per il codice è in ogni caso necessaria, sia essa esplicita come nel caso dei codici algebrici o implicita come in quelli convolutivi. Una struttura consente infatti di accedere direttamente a tutte le parole di codice per mezzo di semplici operazioni algebriche, e quindi di codificare e decodificare i messaggi evitando pesanti ricerche su dizionari di cardinalità esponenziale. Il fatto che i codici correttori promessi dai teoremi di Shannon non siano stati ancora individuati deriva proprio dalla circostanza che essi, oltre che essere "ottimi", devono essere anche "strutturati", e ciò proprio per rendere possibili una codifica e una decodifica efficienti. La teoria (algebrica) dei codici correttori è una disciplina vastissima, e la ricerca in questo settore è ancora fervente. In questa sede ci limiteremo a fornire una descrizione dei codici più importanti, tutti più o meno riconducibili alla sottoclasse dei *codici lineari*, associati alla struttura di *spazio vettoriale*.

5.2 Codice *B-LV* di Huffman

Riprendiamo l'analisi del problema della codifica di sorgente, anticipato nella sezione 3.4, che si esplicita nella ricerca di un codice univocamente decodificabile e ottimo, cioè con un tasso di codifica che sia minimo nella classe cui appartiene il codice. In questa prima sezione ci occupiamo di codici *B-LV* nelle ipotesi di stazionarietà e assenza di memoria, per i quali la ricerca del codice ottimo può essere fatta limitandosi ai codici a prefisso.

Assegnato l'alfabeto $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ della sorgente e la corrispondente distribuzione di probabiltà $P = \{p_1, p_2, \ldots, p_K\}$, si cerca un codice u.d. ottimo $\mathcal{C}^*$, di tipo *B-LV*, su un alfabeto secondario D-ario $\mathcal{B} = \{b_1, b_2, \ldots, b_D\}$, avente K parole di codice $\mathcal{W} = \{w_1, w_2, \ldots, w_K\}$ di lunghezze $\mathcal{L} = \{l_1, l_2, \ldots, l_K\}$; il tasso $R(\mathcal{C}^*)$ deve essere il minimo sulla classe di tutti i codici $\mathcal{C}$ per la sorgente $\mathcal{S}$

$$R(\mathcal{C}^*) \leq R(\mathcal{C}) \qquad \forall \mathcal{C} \tag{5.1}$$

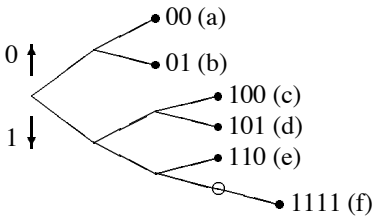
il che equivale a richiedere la minimazione della lunghezza media $E[L]$ nel caso in cui si effettui la codifica sulle singole lettere di $\mathcal{A}$.

Un primo tentativo di risoluzione del problema lo abbiamo anticipato implicitamente nella sezione 3.4.7, individuando la limitazione superiore per il teorema di Shannon (3.14); esso corrisponde al famoso

Codice di Shannon-Fano. A ogni lettera a_i di probabilità p_i si associ una parola di codice di lunghezza $l_i = \lceil -\log_D p_i \rceil$. La costruzione della parola può essere fatta col procedimento del teorema 3.10.

Esempio 5.1. Consideriamo l'alfabeto $\mathcal{A} = \{a, b, c, d, e, f\}$ e la d.p. P indotta dalla normalizzazione degli interi $\{4, 4, 3, 2, 2, 1\}$, cioè $P = \{4/16, 4/16, 3/16, 2/16, 2/16, 1/16\}$, con $D = 2$. Per le lunghezze delle parole di codice si ottiene

$$\begin{aligned}
l_a &= \lceil -\log \tfrac{4}{16} \rceil = \lceil 2 \rceil = 2 \\
l_b &= \lceil -\log \tfrac{4}{16} \rceil = 2 \\
l_c &= \lceil \log 16 - \log 3 \rceil = \lceil 2.415 \rceil = 3 \\
l_d &= \lceil \log 16 - \log 2 \rceil = \lceil 3 \rceil = 3 \\
l_e &= \lceil \log 16 - \log 2 \rceil = \lceil 3 \rceil = 3 \\
l_f &= \lceil \log 16 - \log 1 \rceil = \lceil 4 \rceil = 4
\end{aligned}$$



Poiché le parole di codice non costituiscono una famiglia esauriente (1110 è una via di fuga), la disuguaglianza di McMillan-Kraft vale in senso stretto. Per la lunghezza media si ottiene un valore pari a 41/16, che non può essere ottimo poiché il nodo 1111 potrebbe essere ridotto a 111 senza pregiudicare l'univoca decodificabilità. ◯

Si è visto che le lunghezze l_i soddisfano la disuguaglianza di Kraft e che $E[L_{SF}] < H_D(P) + 1$ (3.66). Nonostante la lunghezza media del codice di Shannon-Fano sia inferiore alla più stretta tra le limitazioni superiori possibili (si ricordi l'osservazione 3.8), essa non è tuttavia ottima. Se prendiamo infatti una sorgente binaria con alfabeto $\mathcal{A} = (a, b)$ e una d.p. $P = (1 - \epsilon, \epsilon)$, si ottiene $l(a) = \lceil -\log(1-\epsilon) \rceil = 1$ e $l(b) = \lceil -\log(\epsilon) \rceil \to +\infty$ quando $\epsilon \to 0$, mentre il codice ottimo ha evidentemente lunghezze $l(a) = 1$, $l(b) = 1$.

Un secondo tentativo di per individuare un codice ottimo lo si ebbe successivamente col codice di *Fano*:

Codice di Fano. Si consideri l'insieme $\mathcal{A}$ e si effettui una partizione in due sottinsiemi, A_0 e A_1, in modo tale che la differenza tra la probabilità cumulata dalle lettere di A_0 e quella cumulata dalle lettere di A_1 sia minima: min $(Pr(A_0) - Pr(A_1))$. Si ripeta il procedimento per A_0 (che si partiziona in A_{00} e A_{01}), A_1 (A_{10} e A_{11}) e per tutte le loro sottopartizioni $A_{i...j}$ finché tutte le sottopartizioni rimangono costituite da una sola lettera di $\mathcal{A}$. Attribuendo 0 a tutte le sottopartizioni con $j = 0$ e 1 a quelle con $j = 1$ si ottiene una parola di codice per ciascuna lettera di $\mathcal{A}$.

Esempio 5.2. Consideriamo lo stesso alfabeto e la stessa d.p. dell'esempio 5.1. La codifica avviene in tre passi, secondo quanto riportato dalla figura 5.1. Si

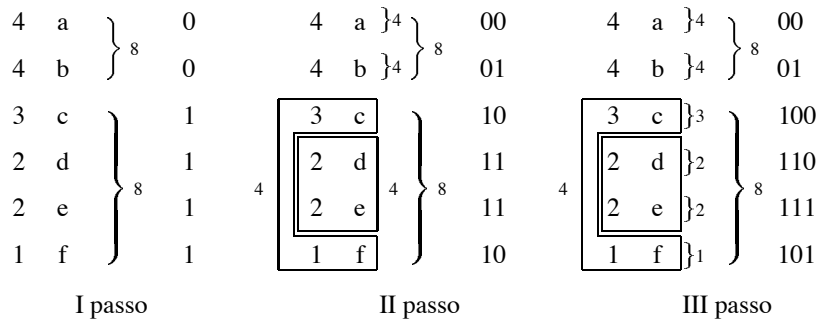


Figura 5.1 Codifica di Fano.

noti che nel terzo passo la suddivisione che separa la f dalla c è sbilanciata. La lunghezza media del codice è pari a 40/16, ed è comunque lievemente inferiore di quella ottenuta col codice di Shannon-Fano. ◯

L'idea del codice è quella di caricare ciascun bit con la massima informazione possibile, il che corrisponde a un bilanciamento delle probabilità $(Pr(A_{i...0}),$

$Pr(A_{i\ldots 1})$) il più possibile uniforme. Si noti che nel nostro caso si sottintende che sia $D = 2$; nel caso generale nulla cambia se non il fatto che la partizione deve comprendere D sottinsiemi. Il codice di Fano funziona bene finché è possibile trovare partizioni sufficientemente bilanciate in probabilità; tuttavia non è detto che non ci si imbatta in situazioni in cui si è obbligati ad accettare alla fine un forte squilibrio. Lo sbilanciamento finale che può accadere induce a pensare che una scelta non ottimale a un passo precedente potrebbe in qualche caso predisporre le partizioni correnti in modo globalmente più favorevole. In altre parole l'algoritmo di Fano ha una ottimalità *locale* che non implica però quella *globale*. Neanche il codice di Fano è dunque ottimo.

Per costruire un codice ottimo bisogna premettere alcuni risultati intermedi, che enunciamo sotto forma di lemma

Lemma 5.1. *Senza perdita di generalità si supponga che sia* $p_1 \geq p_2 \geq \ldots \geq p_K$ *(altrimenti si possono permutare le lettere dell'alfabeto in modo da soddisfare la richiesta); allora esiste un codice ottimo a prefisso per il quale:*

1. *Se* $p_j > p_K$ *si ha* $l_j \leq l_K$ $(j = 1, 2, \ldots, K)$ *(e dunque* $l_1 \leq l_2 \leq \ldots \leq l_K$*).*
2. *Se* $l_K = |w_K|$ *è la lunghezza massimale, esiste un'altra parola* w_j *che ha lunghezza* l_K*;* w_j *differisce da* w_K *solo nell'ultima posizione.*

Dim.

1. Sia $\mathcal{C}^*$ un codice ottimo assegnato. Se scambiamo tra loro le parole w_k e w_j relative alle lettere a_k e a_j otteniamo un codice $\mathcal{C}'$. Poiché $\mathcal{C}^*$ è ottimo, deve essere $E[L(\mathcal{C}')] - E[L(\mathcal{C}^*)] \geq 0$. Calcoliamo la differenza tra le due lunghezze medie

$$\begin{aligned} E[L(\mathcal{C}')] - E[L(\mathcal{C}^*)] &= (p_K l_j + p_j l_K) - (p_j l_j + p_K l_K) = \\ &= p_K (l_j - l_K) - p_j (l_j - l_K) = \\ &= (l_K - l_j)(p_j - p_K) \geq 0 \end{aligned} \tag{5.2}$$

   e poiché $p_j - p_K > 0$ per ipotesi, deve essere $l_K - l_j \geq 0$. Dunque lo stesso $\mathcal{C}^*$ soddisfa la condizione. Poiché il ragionamento può essere ripetuto per qualunque coppia di lettere si ricava che $l_1 \leq l_2 \leq \ldots \leq l_K$

2. Poiché $\mathcal{C}^*$ è u.d. si può suppore che sia a prefisso. Se non esistesse w_j con la stessa lunghezza massimale di w_K, si potrebbe elidere l'ultima lettera di w_K senza violare la proprietà del prefisso, accorciando un codice già ottimo. Le due parole hanno dunque entrambe lunghezza massimale. Senza perdita di generalità, poiché w_K deve possedere un gemello di lunghezza massimale (altrimenti si potrebbe acciorciare il codice), si può pensare che sia $j = K - 1$.

□

La costruzione di $\mathcal{C}^*$ si effettua dunque facilmente se disponiamo delle parole $w_1, w_2, \ldots, w_{K-2}, \vec{w}_{K-1}$ dove $\vec{w}_{K-1}$ è costituito dalle prime $l_K - 1$ lettere di w_K; aggiungendo a queste "0" si ottiene w_{K-1}, mentre aggiungendo "1" si ottiene w_K (nel caso binario; per il caso generale si veda più avanti). In altre parole al posto della sorgente $\mathcal{S} = \mathcal{S}^{(K)}$ possiamo considerare una sorgente *ridotta* $\mathcal{S}^{(K-1)} = \left\{a_1, a_2, \ldots, a_{K-2}, \vec{a}_{K-1}\right\}$, di $K-1$ lettere, in cui la lettera $\vec{a}_{K-1}$ di $\mathcal{S}^{(K-1)}$ esce quando $\mathcal{S}^{(K)}$ emette a_{K-1} *oppure* a_K. Per la sorgente ridotta valgono le seguenti relazioni

$$\begin{aligned} p(\vec{a}_{K-1}) &= \vec{p}_{K-1} = p_{K-1} + p_K \\ l(\vec{a}_{K-1}) &= \vec{l}_{K-1} = l_K - 1 \\ w_{K-1} &= \vec{w}_{K-1}0 \qquad w_K = \vec{w}_{K-1}1 \end{aligned} \tag{5.3}$$

Pertanto, se si dispone di un codice a prefisso $\mathcal{C}^{(K-1)}$ per $\mathcal{S}^{(K-1)}$ se ne ricava subito uno ($\mathcal{C}^{(K)}$) per $\mathcal{S}^{(K)}$, aggiungendo i suffissi "0" e "1" alla parola $w_{K-1} = \vec{w}_{K-1}$. Rimane ora da stabilire il legame tra i due codici.

Teorema 5.1. *Se il codice a prefisso $\mathcal{C}^{(K-1)}$ è ottimo per $\mathcal{S}^{(K-1)}$ allora $\mathcal{C}^{(K)}$ ricavato da $\mathcal{C}^{(K-1)}$ aggiungendo come suffisso "0" e "1" a $\vec{w}_{K-1}$ è ottimo per $\mathcal{S}^{(K)}$.*

Dim. Mettiamo in relazione le lunghezze medie delle parole di codice relative alle due sorgenti

$$\begin{aligned} E[L(\mathcal{S}^{(K)})] &= \sum_{i=1}^{K} p_i l_i = \sum_{i=1}^{K-2} p_i l_i + (p_{K-1} + p_K)\left(\vec{l}_{K-1} + 1\right) = \\ &= \sum_{i=1}^{K-2} p_i l_i + \vec{p}_{K-1}\left(\vec{l}_{K-1} + 1\right) = E[L(\mathcal{S}^{(K-1)})] + \vec{p}_{K-1} = \\ &= E[L(\mathcal{S}^{(K-1)})] + p_{K-1} + p_K \end{aligned}$$

La differenza tra le due lunghezze medie vale $p_{K-1} + p_K$, che è una costante non dipendente dal codice usato; il valor minimo per $E[L(\mathcal{S}^{(K)})]$ lo si trova dunque in corrispondenza del minimo di $E[L(\mathcal{S}^{(K-1)})]$. □

Un codice ottimo per $\mathcal{S}^{(K)}$ si trova allora postponendo "0" e "1" alla parola di codice più lunga relativa al codice ottimo per $\mathcal{S}^{(K-1)}$. In altre parole il problema dell'individuazione di un codice ottimo per $\mathcal{S}^{(K)}$ è stato ridotto a quello dell'individuazione del codice ottimo per $\mathcal{S}^{(K-1)}$. Naturalmente il procedimento può essere iterato finché alla sorgente ridotta rimangono solo due lettere

$\mathcal{S}^{(2)} = \left\{\vec{a}_1, \vec{a}_2\right\}$, per le quali il codice ottimo è evidentemente $\mathcal{C}^{(2)} = \{0, 1\}$. Il punto chiave dell'algoritmo è quello di accorpare sempre le lettere caratterizzate dalle due probabilità più basse. Le parole di codice si costruiscono a partire dall'attribuzione di "0" e "1" alle lettere di $\mathcal{S}^{(2)}$ e concatenando le cifre man mano che, all'atto dell'estensione, si passa da $\vec{w}_j$ a $\vec{w}_j 0$ e $\vec{w}_j 1$.

Algoritmo di Huffman. Assegnata la sorgente $\mathcal{S} = \mathcal{S}^{(K)}$

1. si accorpino le due lettere meno probabili, ottenendo la sorgente ridotta $\mathcal{S}^{(K-1)}$ (per la quale valgono le relazioni (5.3));
2. si riordinino le lettere di $\mathcal{S}^{(K-1)}$;
3. si iteri il procedimento finché si ottiene $\mathcal{S}^{(2)}$;
4. si attribuiscano le due parole "0" e "1" alle lettere di $\mathcal{S}^{(2)}$ e si ripercorra il procedimento all'indietro, estendendo successivamente la sorgente, e concatenando uno "0" e un "1" a ciascuna lettera derivante da una precedente riduzione.

In forza dei teoremi 5.1 e 3.10 il codice di Huffman è *ottimo* all'interno della classe dei codici *B-LV* (a prefisso e non a prefisso), e dunque

$$R(\mathcal{C}^{Huff}) \leq R(\mathcal{C}) \qquad \forall \mathcal{C} \tag{5.4}$$

Si tenga conto del fatto che dal punto di vista operativo la lunghezza media di un codice può essere dedotta direttamente dall'albero del codice, così come attestato dal seguente

Lemma 5.2. *La lunghezza media dei parole di codice (messaggi) associati a un albero di codice è data dalla somma delle probabilità dei nodi interni (di tutti i nodi esclusa la radice).*

Dim. Si tenga conto che il contributo $p_i l_i$ della lunghezza media può essere scritto come $p_i + p_i + \ldots + p_i$ (l_i volte). Percorrendo ora l'albero a partire dalla radice seguendo il percorso indotto da w_i possiamo depositare su ciscuno degli l_i nodi che incontriamo (compresa la radice) un contributo di probabilità paria p_i. Iterando la stessa operazione per tutti i valori di i si ottiene la tesi. □

Esempio 5.3. Consideriamo nuovamente l'alfabeto $\mathcal{A} = \{a, b, c, d, e, f\}$ e la d.p. P indotta dalla normalizzazione degli interi $\{4, 4, 3, 2, 2, 1\}$, cioè $P = \{4/16, 4/16, 3/16, 2/16, 2/16, 1/16\}$, con $D = 2$. L'applicazione dell'algoritmo di Huffman comporta la riduzione successiva della sorgente secondo lo schema della figura 5.2 La procedura di riduzione può essere semplificata lavorando direttamente sull'albero, così come illustrato nella figura 5.3. Ciò consente anche di ricavare immediatamente le parole di codice. La lunghezza media che si ottiene, sommando le probabilità di tutti i nodi interni secondo il lemma (5.2), è pari a 40/16. ◯

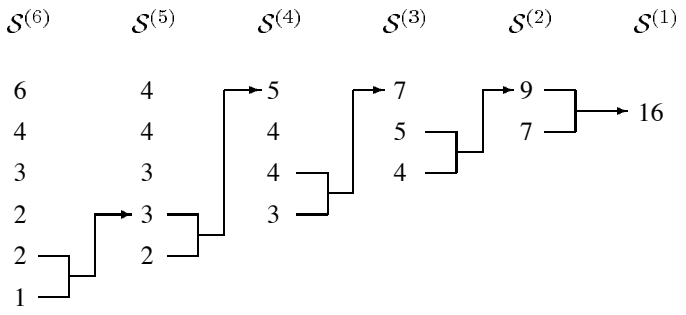


Figura 5.2 Riduzione della sorgente per l'effettuazione dell'algoritmo di Huffman.

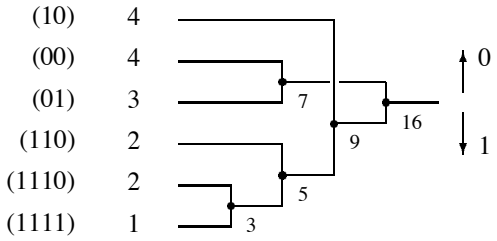


Figura 5.3 L'albero del codice di Huffman per l'esempio di figura 5.2.

Oss. 5.1. La struttura dell'albero associato al codice ottimo potrebbe non essere unica; ciò accade ogniqualvolta si è di fronte a più scelte equivalenti nei nodi da accorpare. Nell'esempio di figura 5.4, dopo aver riunito i primi due nodi (1+1=2) si hanno due modi distinti per realizzare la fusione del tipo 2+1=3. Le due opzioni portano a due alberi con diversa struttura, ma con uguale lunghezza media per il codice associato. Daremo di questo fatto un'interpretazione geometrica nel paragrafo 5.5.1.

Nel caso in cui l'alfabeto secondario non sia binario, cioè $D > 2$, la struttura dell'algoritmo non cambia, salvo che ad ogni riduzione bisogna aggregare le D lettere meno probabili. L'unico problema che si presenta deriva dal fatto che un albero ottimo deve aver saturati tutti i nodi di livelli non massimali, altrimenti si potrebbe accorciare la lunghezza media del codice contravvenendo l'ottimalità. Applicando l'algoritmo descritto sopra, questo non succede quando la cardinalità K dell'alfabeto coincide con la cardinalità di un albero completo D-ario

$$K = D + j(D-1) \tag{5.5}$$

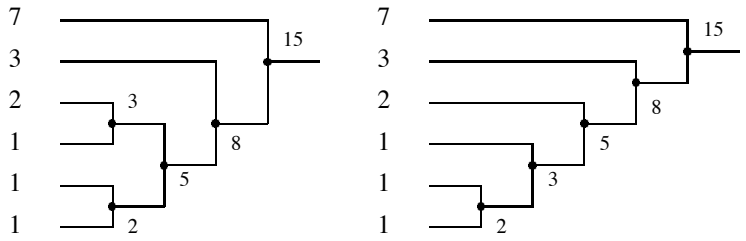


Figura 5.4 Alberi di Huffman strutturalmente diversi, ma con la stessa lunghezza media per il codice associato.

per qualche valore di j (si veda anche il punto 4 della proposizione (3.2). In questo caso si possono aggregare le lettere (foglie) sempre a D a D trovando D foglie anche all'ultima aggregazione. Se la (5.5) non sussiste, (tanto K che D sono assegnati a priori) si può procedere in due modi distinti, o aggregando all'atto della prima riduzione un numero di lettere $d < D$, in modo tale che all'ultima riduzione ci siano sempre D lettere disponibili, oppure aggiungendo $\Delta = D - d$ lettere fittizie, a probabilità zero, in modo tale che la cardinalità soddisfi la (5.5).

Per calcolare d supponiamo che m sia il più piccolo intero tale che

$$T(m) = D + m(D-1) \geq K \quad \text{cioè} \quad K + \Delta = D + m(D-1) \tag{5.6}$$

e dunque bisogna aggiungere Δ lettere fittizie per saturare l'albero. Se il codice è ottimo deve essere $\Delta \leq D - 2$, altrimenti si potrebbe accorciare il ramo che contiene l'unica foglia residua senza perdere la proprietà del prefisso, migliorando un codice già ottimo. Essendo anche $\Delta \geq 0$ si ottiene $0 \leq D-2-\Delta \leq D-2$, che tenuto conto della 5.6 porta alla relazione

$$K - 2 = m(D-1) + D - 2 - \Delta$$

e quindi $D-2-\Delta$ è il resto della divisione di $K-2$ per $D-1$, che indichiamo con $R_{D-1}(K-2)$. Dunque

$$\begin{aligned} \Delta &= D - 2 - R_{D-1}(K-2) \\ d &= R_{D-1}(K-2) + 2 \end{aligned} \tag{5.7}$$

Esempio 5.4. Consideriamo la d.p. indotta dagli interi $\{6, 4, 3, 2, 2, 1\}$ con $D = 3$. Se procediamo direttamente all'aggregazione delle prime tre lettere meno probabili otteniamo l'albero di sinistra; esso non è ottimale in quanto è stata sprecata una parola di codice di lunghezza 1 (la parola (2)) a favore di una di lunghezza maggiore. Il controllo sulla cardinalità richiede l'introduzione di $D - 2 - R_{D-1}(K-2) = 3 - 2 - R_2(4) = 1$ lettera fittizia, oppure di raggruppare al primo passo solo $R_{D-1}(K-2) + 2 = R_2(4) + 2 = 2$ lettere, in modo che l'ultima aggregazione sia satura (si veda figura 5.5). ◯

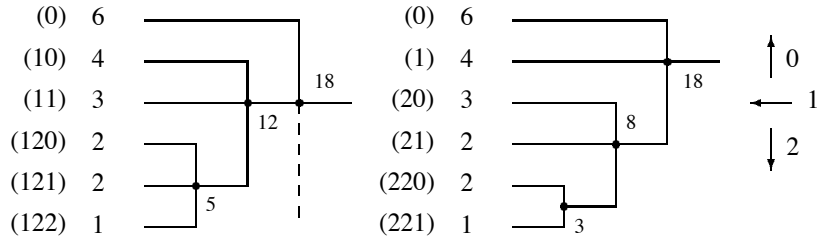


Figura 5.5 Codifica ternaria di Huffman.

A margine possiamo osservare che l'ottimalità del codice di Huffman consente di rendere minimo il tasso, ma non necessariamente di raggiungere l'entropia, se non che con un procedimento asintotico mediante codifica delle estensioni n-esime della sorgente (vedi (3.68)). Come appare infatti evidente dalla (3.65), l'uguaglianza a zero della ridondanza implica che le probabilità si possano esprimere tutte come $p_i = D^{-l_i}$, cioè come esponenti interi negativi di D, mentre di solito esse sono assegnate a priori. In generale la ridondanza di un codice ottimo sarà allora strettamente maggiore di zero, ma minore di 1 (vedi 3.66). Si è studiato inoltre in modo approfondito il rapporto fra d.p. della sorgente e ridondanza indotta per il codice di Huffman; per i dettagli e la bibliografia rimandiamo a [15, 24].

Poiché un codificatore di sorgente ottimo effettua una compressione massimale dei dati, è ragionevole attendersi di non poter effettuare ulteriori compressioni. Come noto ciò accade solo quando la d.p. è uniforme, e dunque l'uscita del codificatore dev'essere assimilabile a quella di una sorgente D-aria quasi uniforme. Infatti, se $H(CS)$ è l'entropia all'uscita del codificatore (che però non fornisce più un processo stazionario e senza memoria; tuttavia si ricordi la (3.8)). Se vogliamo che non vada persa informazione (codice u.d.) deve essere

$$E[L]H(CS) \geq H(\mathcal{S}) \qquad \text{cioè} \qquad E[L] \geq \frac{H(\mathcal{S})}{H(CS)} \tag{5.8}$$

dalla quale si evince che, essendo $H(\mathcal{S})$ bloccata, per effettuare una buona codifica di sorgente bisogna rendere massima la $H(Cod)$. Ciò succede appunto quando la d.p. osservata in uscita è prossima a quella uniforma, e $H(Cod)$ si avvicina alla sua limitazione superiore, cioè $\log D$.

Riprenderemo l'analisi del codice di Huffman nel capitolo dedicato alla codifica adattativa.

5.3 Codice *LV-B* di Tunstall

Nella codifica *LV-B* la sequenza primaria uscente dalla sorgente $\mathcal{S}$ viene segmentata in una successione di messaggi a lunghezza variabile. A ciascun messaggio m_i appartenente a una famiglia completa $\mathcal{M}(j)$ si associa una parola di codice a lunghezza costante $l = \lceil \log_D T(j) \rceil$, dove $T(j) = K + j(K-1)$ è la cardinalità della famiglia $\mathcal{M}(j)$, che corrisponde al numero di foglie di un albero completo K-ario. Il caso della codifica *LV-B* è in un certo senso speculare rispetto a quello *B-LV*. L'ottimalità del tasso (3.69) richiede infatti di render minimo il rapporto

$$R(\mathcal{C}_j^*) = \frac{l}{E[N_j]} = \frac{\lceil \log_D T(j) \rceil}{\sum_{i=1}^{T(j)} q_i n_i} \tag{5.9}$$

e dunque di massimare il denominatore, cioè la lunghezza media dei messaggi che escono dalla sorgente, visto che il numeratore dipende solo dal numero di estensioni fatte. Il problema della codifica ottima a prefisso per il caso *LV-B* può esprimersi nel seguente modo: partendo dalla famiglia minima $\mathcal{M}(0) = \mathcal{A}$ effettuare j estensioni in modo tale che il tasso (5.9) sia minimo.

S'intuisce che il punto critico è la scelta del nodo da estendere secondo il procedimento di estensione di famiglie complete (3.58), descritto nella definizione 3.12. Inoltre l'eventuale ottimalità investe solo la classe dei codici a prefisso, mancando per il caso *LV-B* l'analogo del teorema (3.10). Se riprendiamo il lemma (5.2), possiamo scrivere

$$E[N_{j+1}] = E[N_j] + q_j^* = \sum_{i=0}^{j-1} q_i^* \tag{5.10}$$

dove q_i^* è la successione dei nodi che sono stati estesi. Intuitivamente, per rendere massimo (5.10) bisognerebbe scegliere ad ogni estensione il nodo più probabile tra quelli disponibili, anche se ovviamente non è detto che tale approccio ad ottimalità locale comporti la minimazione *globale* della (5.10). In realtà la scelta del nodo a probabilità massima comporta effettivamente l'ottimalità globale. Introduciamo allora la seguente regola per l'estensione

Algoritmo di Tunstall. A partire dalla famiglia completa $\mathcal{M}(0) = \mathcal{A}$ si effettuino j estensioni scegliendo sempre il nodo più probabile.

Teorema 5.2. *Assegnato j, numero di estensioni, l'albero del codice di Tunstall ha una lunghezza media dei messaggi $E[N_j]$ che è massima su tutti gli alberi completi.*

Dim. Proveremo che un albero completo $\mathcal{A}_C$ non può essere ottimo se non è di Tunstall. Per far questo ricostruiamo l'albero $\mathcal{A}_C$ disponendo i nodi estesi in ordine per probabilità non crescenti. Sia i la posizione della prima estensione per la quale viene violata la regola di Tunstall: $Pr(\nu_i) < Pr(\nu_i^T)$, dove ν_i^T è il nodo che sarebbe stato esteso seguendo l'algoritmo di Tunstall, che è dunque disponibile. Poichè i nodi relativi a estensioni successive sono a probabilità non crescente, la foglia ν_i^T che è stata scartata non potrà più essere usata come nodo da estendere nella costruzione di $\mathcal{A}_C$, e rimarrà per sempre nodo terminale. A questo punto, se stacchiamo il sottoalbero radicato in $Pr(\nu_i)$ e lo innestiamo sul nodo ν_i^T otteniamo un altro albero completo, la cui lunghezza media è maggiore di quella di $\mathcal{A}_C$, creando una contraddizione. □

Oss. *5.2*. I messaggi genarati secondo l'algoritmo di Tunstall hanno una struttura asintoticamente tipica in composizione, nel senso che quelli tipici cumulano una probabilità asintoticamente unitaria. Ciò significa che la loro composizione riflette quella indotta dalla d.p. della sorgente [32]

Esempio 5.5. Si consideri la d.p. $\{0.5, 0.3, 0.2\}$. Le successive estensioni alla Tunstall fino a $j = 3$ portano all'albero di figura 5.6 dal quale si ricavano

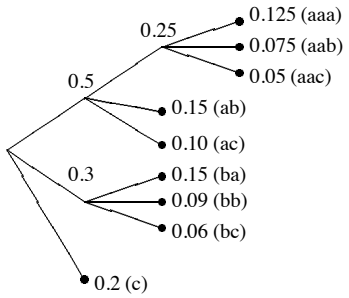


Figura 5.6 Estensioni alla Tunstall per la d.p. $\{0.5, 0.3, 0.2\}$.

$$\mathcal{M}(3) = \{aaa, aab, aac, ab, ac, ba, bb, bc, c\}$$
$$Q(3) = \{0.125, 0.075, 0.05, 0.15, 0.10, 0.15, 0.09, 0.06, 0.2\}$$
$$E[N_3] = 1 + 0.3 + 0.5 + 0.25 = 2.05$$
$$R_3 = \lceil 9 \rceil / 2.05 = 1.95$$

○

L'algoritmo di Tunstall non è ottimo all'interno della classe dei codici non a prefisso, cioè dotati di *ritardo di codifica*. L'esempio seguente illustra questo fatto.

Esempio 5.6. Consideriamo le seguenti due famiglie di messaggi: $\mathcal{M}^T(1) = \{aa, ab, ac, b, c\}$ che è di Tunstall, e $\mathcal{M}^e = \{aaa, aa, a, b, c\}$ che è esauriente, ma non a prefisso (e dunque affetta da ritardo di codifica). Nella codifica di $\mathcal{M}^e$ segmentiamo sempre il prefisso a lunghezza massima (p.es. aaa e non aa/a). Per le due lunghezze medie dei messaggi si ricava rispettivamente

$$\begin{aligned}\bar{n}^T &= [p(aa) + p(ab) + p(ac)]2 + p(b) + p(c) = 1 + p(a)\\ \bar{n}^e &= 3p^3(a) + 2p^2(a)[1 - p(a)] + p(a)[1 - p(a)] + [1 - p(a)] =\\ &= p^3(a) + p^2(a) + 1\end{aligned}$$

L'andamento delle due funzioni è rappresentato in figura 5.7, e si vede che per

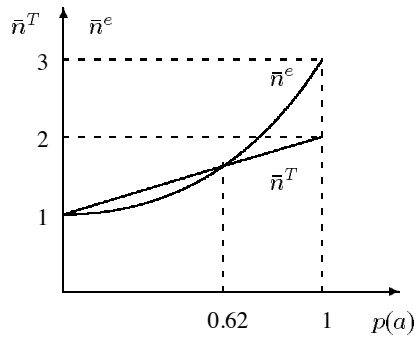


Figura 5.7 Andamento della lunghezza madia dei messaggi per un codice non a prefisso.

$p(a) > (\sqrt{5} - 1)/2 \approx 0.62$ si ha $\bar{n}^e > \bar{n}^T$, e dunque il tasso del codice con ritardo di codifica è migliore di quello di Tunstall. ◯

Per completare l'analisi rimane ora da studiare il comportamento asintotico del tasso di un codice di Tunstall. A tal scopo è opportuno premettere un lemma

Lemma 5.3. *Sia* $Q(j) = \{q_1, q_2, \ldots, q_{T(j)}\}$ *la d.p. dei messaggi (foglie) relativi a un albero di Tunstall dopo* j *estensioni. Definito come* $q_j^- = \min_i q_i$ *e* $q_j^+ = \max_i q_i$ *si ha*

$$\frac{q_j^+}{q_j^-} \leq \frac{1}{p^-} \tag{5.11}$$

dove p^- *è la probabilità minima in* P.

Dim. Si fa per induzione. Se $j = 0$ si ha $p^- = q_0^- = \min_i p_i$ e la relazione è vera in quanto $q_0^+ \leq 1$. Supponiamo che la relazione valga per j e proviamola per $j + 1$. Usando l'algoritmo di Tunstall le probabilità dei nodi scelti successivamente per le estensioni sono non crescenti, e dunque $q_{j+1}^+ \leq q_j^+$, poiché il nodo a probabilità massima o viene coinvolto nell'estensione, e vale la disuguaglianza stretta, oppure ce n'è più d'uno a probabilità massima, e allora vale l'uguaglianza. Il nodo a probabilità minima rimane invece quasi sempre invariato nel passaggio da j a $j + 1$, e dunque $q_{j+1}^- = q_j^-$, a meno che q_{j+1}^- non derivi direttamente dal processo di estensione, e dunque $q_{j+1}^- = q_j^+ p^-$. Nel primo caso si ha

$$\frac{q_{j+1}^+}{q_{j+1}^-} \leq \frac{q_j^+}{q_j^-} \leq \frac{1}{p^-}$$

dove l'ultima disuguaglianza deriva dall'ipotesi induttiva. Per il secondo caso si ha invece

$$\frac{q_{j+1}^+}{q_{j+1}^-} \leq \frac{q_j^+}{q_{j+1}^-} = \frac{q_j^+}{q_j^+ p^-} = \frac{1}{p^-}$$

e la disuguaglianza è soddifatta in ogni caso. □

Oss. *5.3*. Il lemma appena visto ha una controparte in termini di lunghezze massimali e minimali $n^+(j)$ e $n^-(j)$ dei messaggi dell'albero di codice; si potrebbe infatti dimostrare [31] che

$$\lim_{j \to \infty} \frac{n^+(j)}{n^-(j)} = \frac{\log p^-}{\log p^+}$$

dove $p^+ = \max_i p_i$. L'asimmetria asintotica dell'albero è dunque controllata dallo sbilanciamento della d.p. della sorgente.

Il comportamento asintotico del tasso deve essere studiato cercando per esso una limitazione superiore, come è stato fatto nel caso *B-LV*, verificando se detta limitazione collassa su quella inferiore, che è fissa rispetto al parametro asintotico j. Riprendiamo la (3.72)

$$R_j = \frac{l}{E[N_j]} = \lceil \log_D T(j) \rceil \frac{H_D(P)}{H_D(Q(j))}$$

Per individuare una limitazione superiore bisogna limitare inferiormente l'entropia

$$H_D(Q(j)) = -\sum_{i=1}^{T(j)} q_i \log_D q_i \geq -\sum_{i=1}^{T(j)} q_i \max_{1\leq r\leq T(j)} \log_D q_r =$$
$$= -\max_{1\leq r\leq T(j)} \log_D q_r \overset{(1)}{=} -\log_D q_j^+ \overset{(2)}{\geq} -\log_D \left(\frac{q_j^-}{p^-}\right) =$$
$$= \log_D p^- - \log_D q_j^- \overset{(3)}{\geq} \log_D p^- + \log_D T(j)$$

dove la $^{(1)}$ deriva dal fatto che il logaritmo è una funzione monotona, la $^{(2)}$ dall'applicazione del lemma (5.3) e la $^{(3)}$ dal fatto che $q_j^- \leq \frac{1}{T(j)}$. Sostituendo ricaviamo

$$H_D(P) \leq R_j = \frac{\lceil \log_D T(j) \rceil}{H_D(Q(j))} H_D(P) < \frac{1 + \log_D T(j)}{\log_D p^- + \log_D T(j)} H_D(P)$$

Si noti infine che

$$\lim_{j\to\infty} \frac{1 + \log_D T(j)}{\log_D p^- + \log_D T(j)} = 1 \qquad \text{e dunque} \qquad \lim_{j\to\infty} R_j = H_D(P)$$

Il codice di Tunstall è dunque ottimo nell'ambito dei codici a prefisso, il suo tasso tende asintoticamente all'entropia, tuttavia al finito le sue prestazioni si possono superare usando codici con ritardo di codifica. In quest'ultimo caso il problema dell'individuazione del codice ottimo è ancora aperto.

5.4 Codici *LV-LV*

La codifica *LV-LV*, come già anticipato nella sezione 3.4.9, è il caso più generale possibile di codifica di sorgente. Anche se la variabilità delle lunghezze di messaggi e parole di codice dovrebbe in linea di principio fornire un grado di libertà in più nel progetto del codice, consentendo prestazioni più spinte al finito per quanto riguarda il tasso, in pratica bisogna tener conto del fatto che non c'è indipendenza tra le due codifiche *LV-B* e *B-LV*, poiché la scelta delle estensioni necessarie a costruire la famiglia di messaggi a lunghezza variabile determina implicitamente le prestazioni dell'intero codice. Per rendere minimo il tasso bisogna fare in modo che la distribuzione $Q(j)$ associata ai messaggi sia molto simile a una diadica (per minimare il numeratore del tasso), consentendo nel contempo un valore elevato per la lunghezza media dei messaggi (il che rende massimo il denominatore).

In pratica, una volta stabilita la famiglia di messaggi a lunghezza variabile per il codice *LV-B*, non resta altro da fare se non usare un codice di Huffman per

endere minimo il tasso del successivo codice *B-LV*. Il vero problema è dunque quello di costruire la famiglia $\mathcal{M}$.

Senza entrare nei dettagli, per i quali rimandiamo a [27], ricordiamo solo che a concatenazione di un codice ottimo di Tunstall con un codice ottimo di Huffman porta a un codice asintoticamente ottimo di tipo *LV-LV*, il cui tasso tende quindi all'entropia della sorgente. Per rendersi conto di ciò basta esprimere il tasso in funzione dell'entropia delle foglie dell'albero per la codifica *LV-B*

$$R_j = \frac{E[L_j]}{E[N_j]} = \frac{H_{_D}(Q(j)) + r_j}{E[N_j]} = H_{_D}(P) + \frac{r_j}{1 + \sum_{i=0}^{j-1} q^*(i)} \qquad (5.12)$$

dove $E[N_j] = 1 + \sum_{i=0}^{j-1} q^*(i)$ è la somma delle probabilità dei nodi estesi, che a norma del lemma (5.2) corrisponde alla lunghezza media dei messaggi; r_j è invece la ridondanza al passo j del codice *B-LV* (si ricordi la (3.65)), che per un codice di Huffman è limitata superiormente da 1 per effetto della (3.66).

Per ottenere l'ottimalità è allora sufficiente che nella (5.12) la probabilità dei nodi estesi tenda all'infinito al crescere di n, il che accade di sicuro per il codice di Tunstall visto che in questo caso si sceglie semnpre il nodo più probabile.

Si noti tuttavia che la concatenazione Tunstall-Huffman non porta al codice ottimo al finito, come si può verificare facilmente mediante l'esempio sotto riportato.

Esempio 5.7. Si consideri la sorgente binaria associata alla d.p. $P = \{0.7, 0.3\}$. Eseguendo due estensioni alla Tunstall si perviene alla d.p. $Q(2) = \{0.3, 0.343, 0.147, 0.21\}$, che codificate alla Huffman portano a un tasso pari a 0.91324. Nel secondo caso l'estensione *LV-B* non è più alla Tunstall (si sceglie il nodo 0.21 in luogo di 0.49, che è quello a probabilità massima). Tuttavia il risultato globale è migliore, poiché a seguito di una codifica alla Huffman si ottiene un tasso pari a 0.90052, inferiore a quello precedente (vedi fig. 5.8).

Il problema dell'individuazione del codice ottimo *LV-LV* per famiglie complete è ancora aperto.

5.5 Codifica adattativa

Nelle sorgenti reali, come più volte ricordato, l'ipotesi di stazionarietà del processo è poco verosimile; la d.p. della sorgente può subire variazioni durante il funzionamento della stessa, rendendo non più ottimo per una nuova d.p. Q un codice che era stato progettato per una d.p. P precedente. A ciò si sovrappone il problema della *stima* della d.p. di sorgente: se nel momento in cui una sorgente inizia a funzionare non abbiamo informazioni attendibili sulla sua statistica, dovremo effettuare una serie $\widehat{Q_i}$ di stime man mano che le lettere escono dalla

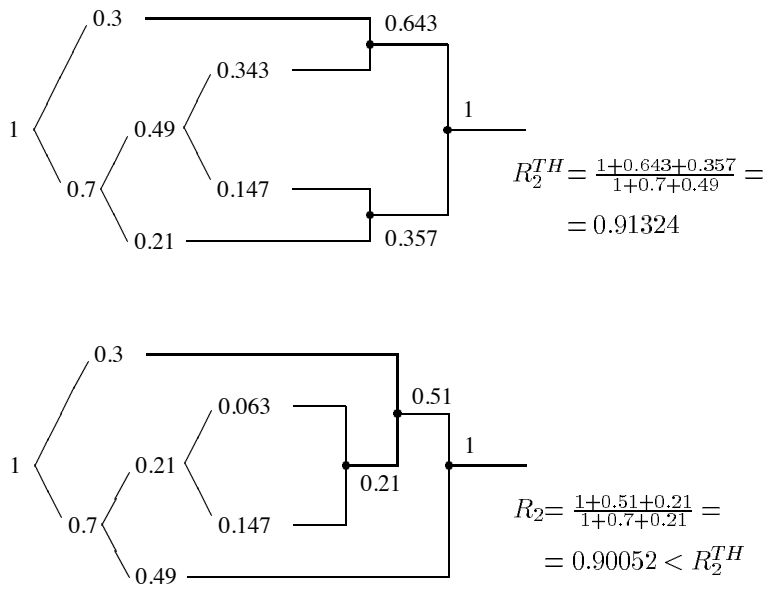


Figura 5.8 Codice *LV-LV* ottenuto come concatenazione Tunstall-Huffman e albero ottimo

sorgente, approssimandoci sempre più alla "vera" d.p. P. Poiché i codici vengono costruiti sulla base delle $\widehat{Q_i}$ è come se fossimo di fronte a una sorgente che varia nel tempo la sua distribuzione di probabilità.

Il primo problema da affrontare è allora quello di valutare la *perdita di ottimalità* che si soffre quando si usa un codice progettato sulla base di una d.p. P per codificare una sorgente che sta funzionando con una d.p. $Q \neq P$. Tale contesto è anche noto col nome di *disadattamento di sorgente*.

A tal riguardo supponiamo di usare un codice di Shannon-Fano per una codifica *B-LV*. Sia $P = \{p_1, p_2, \ldots, p_K\}$ la d.p. vera della sorgente e $Q = \{q_1, q_2, \ldots, q_K\}$ una sua stima sulla quale basare il codice. Siano inoltre

$$l_i(P) = \lceil -\log_D p_i \rceil \qquad \text{e} \qquad l_i(Q) = \lceil -\log_D q_i \rceil$$

rispettivamente le lunghezza delle parole di codice che si potrebbe costruire *conoscendo* la d.p. P e le lunghezze delle parole costruite sulla base della stima Q. Per evitare l'uso della parte intera superiore possiamo anche scrivere

$$l_i(P) = -\log_D p_i + \delta_i(P) \qquad \text{e} \qquad l_i(Q) = -\log_D q_i + \delta_i(Q)$$

con $0 \leq \delta_i < 1$. Vale il seguente teorema

Teorema 5.3. *La perdita asintotica di ottimalità che si accusa codificando alla Shannon-Fano, sulla base di una d.p. Q, una sorgente che sta funzionando con una d.p. P, è asintoticamente pari alla divergenza informazionale $D(P//Q)$ tra P e Q.*

Dim. Calcoliamo la differenza tra le due lunghezze medie

$$E[L(Q)] - E[L(P)] = \sum_{i=1}^{K} p_i l_i(Q) - \sum_{i=1}^{K} p_i l_i(P) =$$
$$= -\sum_{i=1}^{K} p_i \log_D q_i + \sum_{i=1}^{K} p_i \delta_i(Q) + \sum_{i=1}^{K} p_i \log_D p_i - \sum_{i=1}^{K} p_i \delta_i(P) =$$
$$= \sum_{i=1}^{K} p_i \log_D p_i - \sum_{i=1}^{K} p_i \log_D q_i + \Delta = D(P//Q) + \Delta$$

con $\Delta = \sum_{i=1}^{K} p_i \delta_i(Q) - \sum_{i=1}^{K} p_i \delta_i(P)$ e $|\Delta| < 1$ in quanto ottenuto come differenza di termini maggiori di 0 e strettamente minori di 1. Se codifichiamo l'estensione n-esima della sorgente la differenza normalizzata vale

$$\frac{E[L(Q^n)] - E[L(P^n)]}{n} = \frac{D(P^n//Q^n)}{n} + \frac{\Delta}{n} = D(P//Q) + \frac{\Delta}{n} \qquad (5.13)$$

cioè la tesi, tenuto conto che $D(P^n//Q^n) = nD(P//Q)$ quando valgono le ipotesi di stazionarietà e assenza di memoria. □

Oss. 5.4. La (5.13) è stata calcolata usando il codice di Shannon-Fano. Si potrebbe dimostrare, con un procedimento molto più complicato, che tanto per il codice di ottimo *B-LV* di Huffman quanto per quello ottimo *LV-B* di Tunstall si ha la stessa perdita di ottimalità asintotica (si veda [65] e [31]). La divergenza caratterizza dunque la perdita di ottimalità a prescindere dal codice in uso, sempreché questo sia asintoticamente ottimo.

5.5.1 Codifica adattativa geometrica

Dalla (5.13) risulta chiaro che, quando la d.p. vera P comincia a variare, per mantenere l'ottimalità bisogna "inseguire" la P, mantenendo aggiornato il codice sulla successione delle d.p. Q_i che via via prendono il posto della P. In altre parole, per ogni nuova d.p. Q_i dobbiamo ripetere l'algoritmo di Huffman, mantenendo sempre ottimo l'insieme delle lunghezze delle parole di codice. Per

precisare meglio questo aspetto riconsideriamo la *ridondanza del codice* (3.65), limitandoci per semplicità al caso binario

$$r(\mathcal{C}) = E[L(\mathcal{C})] - H(P) = \sum_{i=1}^{K} p_i l_i + \sum_{i=1}^{K} p_i \log p_i =$$
$$= \sum_{i=1}^{K} p_i \log \frac{p_i}{2^{-l_i}} = D(P//E^*) \tag{5.14}$$

dove $E^* = \left\{2^{-l_i}\right\}_{i=1}^{K}$ è una d.p. nella forma *diadica* (si veda l'osservazione 3.6), cioè come esponente negativo di 2, e gli esponenti sono esattamente le lunghezze delle parole di codice. Il codice ottimo di Huffman rende minima la ridondanza, cioè la divergenza informazionale $D(P//E)$, che si può interpretare come una (pseudo)-*distanza* tra la d.p. P e la diadica E (per lo studio sistematico delle proprietà della divergenza come psudo-metrica rimandiamo a [74]). Consideriamo allora lo spazio K-dimensionale $\wp^K$ che corrisponde all'insieme di *tutte le distribuzioni di probabilità K-adiche*, e immaginiamo tutte le possibili diadiche immerse in $\wp^K$. Si noti che le diadiche sono prefissate e non dipendono dal codice (l'unica condizione è che siano esprimibili come potenze intere negative di 2). Fissata P all'interno dello spazio, l'attuazione dell'algoritmo di Huffman comporta l'individuazione, tra tutte le possibili diadiche, di quella che realizza la *minima divergenza* $D(P//E^*)$ (si veda figura 5.9). Questo fatto

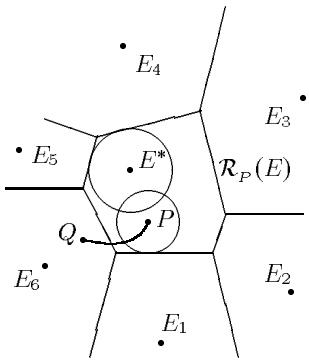


Figura 5.9 Regioni di attrazione associate alle diadiche E_j.

costituisce un'interpretazione *geometrica* della codifica, che merita un'approfondimento. Consideriamo accanto a ogni diadica E l'insieme $\mathcal{R}(E)$ di tutte le d.p. che vengono codificate in modo ottimo (cioè secondo l'algoritmo di Huffman) dalle lunghezze che corrispondono agli esponenti della diadica E. È ovvio che $E \in \mathcal{R}(E)$, in quanto applicando l'algoritmo di Huffman alla E si ottengono come lunghezze delle parole di codice proprio le $\mathcal{L}_P = \{l_1, l_2, \ldots, l_K\}$.

Si potrebbe verificare che l'insieme $\mathcal{R}(E)$ è convesso; esso viene chiamato *regione di attrazione della d.p. E*. Ogni regione di attrazione $\mathcal{R}(E)$ contiene E al proprio interno, e confina con altre regioni di attrazione di diadiche attigue. Supponiamo ora che la d.p. associata alla sorgente cominci a variare a partire da una condizione iniziale P; la variazione può essere interpretata come uno spostamento geometrico all'interno dello spazio $\wp^K$. Se $\mathcal{R}_P(E)$ è la regione di attrazione di partenza, il corrispondente insieme ottimo $\mathcal{L}_P$, ricavato mediante l'algoritmo di Huffman, non dovrà essere variato *fintantoché la nuova d.p. Q rimane all'interno della stessa regione di attrazione*. In tal caso *non è necessario riapplicare l'algoritmo di Huffman* per Q. Il problema è che questa informazione non è disponibile a priori, e bisogna dunque effettuare una qualche textitverifica, che deve però essere più economica in termini computazionali dell'applicazione dell'algoritmo di Huffman.

Criterio di Longo-Galasso. Se $\mathcal{R}_P(E)$ è la regione di attrazione associata alla d.p. iniziale P, si costruiscano le più grandi sfere centrate rispettivamente in E e P e interamente contenute in $\mathcal{R}_P(E)$. Siano ρ_E e ρ_P i raggi delle sfere e Q la nuova d.p. Se

$$D(Q//E) \leq \rho_E \qquad \text{oppure} \qquad D(Q//P) \leq \rho_P$$

allora la d.p. Q sta ancora nella regione di attrazione iniziale e non è necessario aggiornare il codice ottimo.

I raggi delle sfere si ricavano dalle seguenti relazioni

$$\rho_E = -\log\left[1-(3-2\sqrt{2})2^{-l_{MAX}}\right] \;\; \rho_P = -\log\left[1-\min(\sqrt{p_+}-\sqrt{p_-})^2\right] \tag{5.15}$$

dove l_{MAX} è la lunghezza massimale del codice, mentre p_+ e p_- sono le probabilità di nodi che stanno su livelli attigui nell'albero del codice di Huffman (per i dettagli si veda [58]).

Si noti che la verifica esprime una condizione solo sufficiente, poiché può accadere che una d.p. stia al di fuori delle sfere, ma dentro la regione di attrazione. In tal caso il criterio indica la necessità di riapplicare l'algoritmo di Huffman, che fornisce però le stesse lunghezze di prima.

Osserviamo ancora che le regioni di attrazione non costituiscono una *partizione* di $\wp^K$; se infatti un punto P si trova equidistante da due diadiche E_1 e E_2 (a distanza minima), esso appartiene contemporaneamente alle due regioni $\mathcal{R}(E_1)$ e $\mathcal{R}(E_2)$, appartenendo così alla loro *frontiera*. Se il punto è equidistante da tre diadiche, allora esso appartiene all'intersezione tra le tre frontiere e ci sono tre codici distinti di Huffman, strutturalmente diversi, che portano alla minimazione della lunghezza media. L'appartenenza di una d.p. a una frontiera si evidenzia,

nell'esecuzione dell'algoritmo di Huffman, nel momento in cui ci sono *più scelte* per quanto riguarda l'accoppiamento delle due lettere a probabilità minima (vedi figura 5.4).

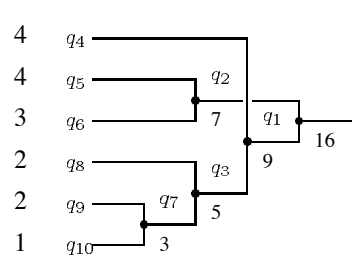
Questo metodo adattativo basa il proprio funzionamento su una caratterizzazione geometrica della codifica che ha un interesse intrinseco; tuttavia, dal punto di vista applicativo l'importanza del metodo è frustrata dalla circostanza che, quando la cardinalità dell'alfabeto diventa rilevante (p.es. a seguito di una codifica dell'estensione n-esima della sorgente), il raggio delle sfere decresce rapidamente, come si verifica subito dalla (5.15). Le diadiche tendono dunque a compattarsi, e basta variare di poco la d.p. iniziale per cadere al di fuori della regione di attrazione, che impone un nuovo calcolo delle lunghezze tramite l'algoritmo di Huffman. Inoltre il volume delle sfere, rapportato a quello totale della regione, diventa percentualmente trascurabile quando $K \to \infty$, e ciò implica una perdita d'efficacia del metodo.

5.5.2 Aggiornamento adattativo dell'albero di codice

La variazione temporale di qualche componente della d.p. P si manifesta durante l'esecuzione dell'algoritmo di Huffman modificando la sequenza delle coppie di lettere che vengono scelte per costruire le sorgenti ridotte. Tale modifica comporta una diversa strutturazione dell'albero di codifica, e dunque un diverso insieme $\mathcal{L}$ delle lunghezze delle parole di codice. Immaginiamo di avere a disposizione l'albero ottimo corrente $\mathfrak{A}_P$ per la d.p. P, e che un passaggio da P a Q comporti la costruzione di un nuovo albero ottimo $\mathfrak{A}_Q$. Invece che eseguire ripetutamente l'algoritmo di Huffman per ogni nuova d.p. acquisita si può pensare di aggiornare direttamente l'albero di codice, radicando un suo sottoalbero su un nodo diverso. Naturalmente se la variazione della d.p. è "piccola", non sarà necessario aggiornare nulla, poiché si rimane all'interno della stessa regione di attrazione e il vecchio albero continua a essere ottimo anche per la nuova d.p.. La caratterizzazione geometrica dell'aggiornamento del codice ottimo, cioè il passaggio da una regione di attrazione all'altra, ha anche una controparte in termini strutturali dell'albero di codice. Introduciamo la seguente

Def. 5.1. *Un albero di un codice binario gode della proprietà dei gemelli se ciascun nodo, eccetto la sorgente, ha un gemello e se i nodi dell'albero possono essere posti in ordine per probabilità decrescenti in modo che ciascun nodo sia adiacente al proprio gemello.*

La figura 5.10 illustra la proprietà. Naturalmente se nodi diversi hanno la stessa probabilità le liste possono essere diverse, e la proprietà vale a maggior ragione. Per il codice di Huffman vale allora il seguente



q_1 q_2	9 7
q_3 q_4	5 4
q_5 q_6	4 3
q_7 q_8	3 2
q_9 q_{10}	2 1

} gemelli

Figura 5.10 Tabella per la verifica della proprietà dei gemelli.

Teorema 5.4. *Un codice binario a prefisso è un codice di Huffman se e solo se il suo albero gode della proprietà dei gemelli.*

Dim. Se l'albero gode della proprietà dei gemelli gli ultimi due elementi della lista (quelli a probabilità minima) sono gemelli, e devono essere nodi terminali, giacché se non lo fossero i loro figli avrebbero probabilità inferiori, contraddicendo il fatto che i genitori sono in fondo alla lista. Questi due nodi corrispondono alle due lettere di sorgente a probabilità minima, e possono essere accoppiati nell'esecuzione del primo passo dell'algoritmo di Huffman. Rimuoviamo ora questi gemelli dall'albero di codice e dalla lista ordinata. L'albero di codice ridotto gode ancora della proprietà dei gemelli, e il procedimento può essere iterato: a ogni passo l'algoritmo di Huffman sceglie come lettere da accoppiare gli elementi che sono gemelli nell'albero di codice. L'albero di codice è dunque un albero ottimo di Huffman.

Viceversa supponiamo che un albero di codice binario sia generato dall'algoritmo di Huffman, e assumiamo che ogniqualvolta l'algoritmo accorpa le due lettere meno probabili, i corrispondenti nodi vengano inseriti come gemelli sulla cima di una lista vuota, mettendo il nodo meno probabile sotto quello più probabile. La lista generata ha chiaramente ciascun nodo adiacente al proprio gemello, e per la verifica della proprietà basta controllare che le probabilità siano ordinate in modo non crescente. Ma questo è banalmente verificato per l'algoritmo di Huffman, poiché durante ciascuna iterazione i due elementi aggiunti alla lista hanno probabilità che sono minori o uguali a quelle relative a nodi che saranno posti successivamente nella lista. □

Il teorema 5.4 caratterizza l'ottimalità di un codice *B-LV* mediante la validità della proprietà dei gemelli. Ciò consente di sfruttare questa proprietà per *tenere aggiornato dinamicamente* l'albero ottimo del codice.

Si abbia a disposizione un albero ottimo associato inizialmente a una d.p. P e la corrispondente tabella con le probabilità dei gemelli. Poiché a ogni nodo dell'albero corrisponde un elemento della tabella, possiamo usare una struttura dati, del tipo di quella illustrata in figura 5.11, costituita da alcune celle di memoria, una per nodo, associate a un contatore . Ogniqualvolta esce una lettera a_i dalla sorgente, il valore dell'i-esimo contatore viene incrementato di 1, e tale incremento si propaga su tutti i nodi che si frappongono tra il nodo della lettera e la radice. Il contatore delle lettere è proporzionale alle frequenza relativa delle stesse, e costituisce una *stima* della probabilità corrispondente, mentre i contatori dei nodi interni all'albero rappresentano le frequenze relative dell'unione di gruppi di lettere, che corrispondono alle lettere delle sorgenti ridotte. Poiché l'albero è quello ottimo deve valere per esso la proprietà dei gemelli, e quindi nella tabella ogni nodo è adiacente al proprio gemello, e i nodi sono ordinati per probabilità decrescenti. Quando il valore dei contatori viene incrementato dalle lettere che via via escono dalla sorgente, l'albero continua a essere ottimo fintantoché vale l'ordinamento indotto dalla proprietà dei gemelli. La struttura dell'albero viene mantenuta mediante alcuni puntatori diretti (PD), che partono dalla coppia di gemelli e puntano al nodo genitore (più precisamente alla coppia di gemelli che contiene il genitore). Per distinguere tra due gemelli si può associare al puntatore uno "0" o un "1". Le lettere di sorgente hanno un'area di memoria a parte e contengono, oltre al contatore, il puntatore verso la coppia di gemelli corrispondenti. Quest'ultimo consente, assieme a tutti gli altri, di attribuire direttamente la parola di codice alla lettera in uscita. Se nell'esempio di figura 5.11 esce la lettera a_1, si può leggere la parola corrispondente partendo da a_1 e seguendo tutti i puntatori fino alla radice; in questo caso la codifica è "00".

Aggiornamento adattativo di Gallager. Ad ogni uscita di una lettera, cui corrisponde un'incremento di tutti i contatori che si incontrano procedendo dalla lettera verso la radice, è necessario verificare se il contenuto di ciascun contatore è sempre minore o uguale a quello del livello immediatamente superiore; se ciò non succede si devono scambiare i due nodi, inducendo lo scambio dei puntatori diretti corrispondenti, in modo da mantenere la struttura ottima dell'albero. Lo scopo dei puntatori inversi PI_0 e PI_1 è quello di localizzare i puntatori diretti evitando una ricerca. Il passaggio da un albero a un altro si realizza commutando due puntatori diretti e i due corrispondenti inversi.

Se per esempio nello schema della figura 5.11 esce tre volte la lettera a_6 s'impone uno scambio tra quest'ultima e a_5 ; in tal caso il contenuto del contatore 1 della 5^a coppia di gemelli viene incrementato a 8, i puntatori diretti e inversi dei due nodi si scambiano, e il contenuto del contatore 1 della 4^a coppia viene portato a 15. Una nuova uscita di a_6 (o anche di a_5) determina infine lo scambio dei

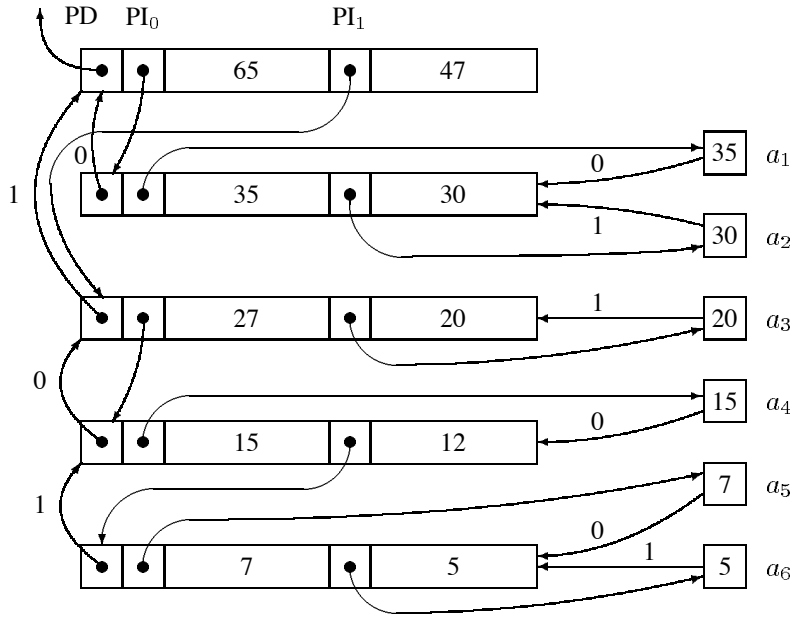


Figura 5.11 Struttura dati per l'aggiornamento adattativo dell'albero di codice.

contenuti dei contatori 0 e 1 della 4^a coppia, inducendo anche un aggiornamento dei rispettivi puntatori.

Il metodo descritto è molto semplice e flessibile, e ha trovato largo impiego nei programmi commerciali per la compattazione dei dati (p.es. la routine *compact* per UNIX; si vedano [54] e [42] per una rassegna dei metodi di compressione impiegati nella pratica). Naturalmente, pur avendo sempre a disposizione le informazioni inerenti la struttura dell'albero ottimo, non è necessario adeguare il codice ad ogni variazione dello stesso, poiché si rischia che le oscillazioni statistiche della sorgente comportino un'aggiornamento continuo e un appesantimento computazionale.

Oss. 5.5. I due procedimenti adattativi visti poc'anzi hanno una controparte anche nell'ambito della codifica ottima *LV-B* di Tunstall. Dalla struttura dell'albero ottimo *LV-B* si può inoltre ricavare una *proprietà dell'ordinamento* che è analoga alla proprietà dei gemelli. Tenuto conto anche dell'osservazione 5.4 sulla perdita asintotica d'ottimalità, si ricava una forte analogia nel comportamento dei due codici ottimi a lunghezza variabile.

Un approccio alternativo alla codifica adattativa può essere fornito dall'impiego delle cosidette *reti neurali*, basate sul modello di *Hopfield*[], che consentono di mutuare complessità computazionale con complessità strutturale della

rete. In tal caso la rete viene alimentata con le frequenze relative delle varie lettere dell'alfabeto e fornisce in tempo virtualmente reale (mediante una computazione di tipo parallelo) le lunghezze delle parole di codice che mantengono l'ottimalità del codice. Come nel caso del codice adattativo di Gallager il codice può essere aggiornato con una frequenza che dipende dalle caratteristiche della sorgente.

5.6 Codici universali

In tutte le considerazioni fatte finora, con riguardo al problema della compressione dei dati, si è fatto sempre riferimento alla statistica della sorgente. La costruzione effettiva delle parole di codice, tanto nel caso *B-LV* che in quello *LV-B*, si è basata sulla conoscenza della distribuzione di probabilità della sorgente (stazionaria e senza memoria), ottenendo un tasso che si approssima asintoticamente all'entropia. Questa dipendenza forte risulta intuitiva, poiché è ben comprensibile che la minimazione del tasso, nel caso dei codici *B-LV*, può essere ottenuta associando alle lettere più probabili le parole di codice più corte (per il caso *LV-B* si devono trovare messaggi più lunghi). Non è dunque scontato che esistano degli schemi di codifica *universali*, che forniscono prestazioni analoghe in termini di tasso, ma *senza* che sia nota la d.p. della sorgente. Nel seguito analizzeremo due esempi, quello del *codice multinomiale* e quello del *codice di Ziv-Lempel*, che sono molto importanti tanto dal punto di vista normativo-concettuale che applicativo.

5.6.1 Codice multinomiale

La costruzione del codice multinomiale sfrutta la struttura e le proprietà dei *tipi esatti*, già incontrati nella sezione 3.3.2. Ricordiamo che l'insieme delle K^n possibili n-ple su un alfabeto K-ario viene partizionato in $\Gamma_n \leq (n+1)^K$ classi di equivalenza, cioè i tipi esatti (o tipi). Ciascun tipo contiene tutte e sole le n-ple che sono l'una permutazione dell'altra. Il tipo esatto è rappresentato dalla sua composizione, cioè dal vettore $(n_1, n_2, \ldots, n_K)$, dove con $n_i = N(x_i/\boldsymbol{x})$ indichiamo il numero di volte in cui la lettera x_i compare nella n-pla $\boldsymbol{x}$. I vincoli indotti sono $0 \leq n_i \leq n$ e naturalmente $\sum_{i=1}^{K} n_i = n$. La cardinalità di ciscun tipo dipende dalla sua composizione, ed è determinata dal coefficiente *multinomiale* della (3.32)

$$|T_{\boldsymbol{x}}| = \frac{n!}{n_1!\, n_2! \ldots n_K!} = \binom{n}{n_1\, n_2 \ldots n_K}$$

da cui il nome del codice.

L'idea di base per attuare la codifica è molto semplice, e consiste nel codificare le "coordinate" dell' ennupla in uscita $\boldsymbol{x}$, indicando prima il tipo cui appartiene e successivamente la sua posizione all'interno del tipo.

Algoritmo di codifica multinomiale. Ciascuna n-pla $\boldsymbol{x}$ viene codificata con una parola a lunghezza variabile

$$\varphi(\boldsymbol{x}) = \varphi_{P} * \varphi_{S}(\boldsymbol{x}) \tag{5.16}$$

costituita da un *prefisso* φ_{P}, a lunghezza costante, che specifica il numero d'ordine (lessicografico) del tipo esatto cui appartiene $\boldsymbol{x}$, e un *suffisso* $\varphi_{S}(\boldsymbol{x})$, a lunghezza variabile, che specifica il numero d'ordine della n-pla all'interno del tipo esatto.

Se $l_{P} = |\varphi_{P}|$ e $l_{S}(\boldsymbol{x}) = |\varphi_{S}(\boldsymbol{x})|$ il vincolo dell'univoca decodificabilità impone le seguenti condizioni sulle lunghezze del prefisso e del suffisso

$$\begin{aligned} D^{l_P} &\geq \Gamma_n & l_{P} &= \lceil \log_{D} \Gamma_n \rceil \\ D^{l_S(\boldsymbol{x})} &\geq |T_{\boldsymbol{x}}| & l_{S}(\boldsymbol{x}) &= \lceil \log_{D} |T_{\boldsymbol{x}}| \rceil \end{aligned} \tag{5.17}$$

Il codice multinomiale è dunque un codice *B-LV*.

Esempio 5.8. Supponiamo che sia $\mathcal{A} = \{a, b\}$ e $\mathcal{B} = \{0, 1\}$, cioè $K = 2$, $D = 2$ con $n = 3$. Ci sono $\Gamma_3 = \binom{n+K-1}{n} = \binom{4}{3} = 4$ tipi esatti, $(3,0), (0,3)$, $(2,1)$ e $(1,2)$. La lunghezza del prefisso è pari a $l_{P} = \lceil \log_2 4 \rceil = 2$.

Tipo	n-pla	$l_{S}(\boldsymbol{x})$	prefisso	suffisso	codifica
(3,0)	aaa	$\lceil \log_2 1 \rceil = 0$	00	-	00
(0,3)	bbb	$\lceil \log_2 1 \rceil = 0$	01	-	01
(2,1)	aab	$\lceil \log_2 3 \rceil = 2$	10	00	1000
	aba		10	01	1001
	baa		10	10	1010
(1,2)	bba	$\lceil \log_2 3 \rceil = 2$	11	00	1100
	bab		11	01	1101
	abb		11	10	1110

L'albero di codice rappresentato in figura 5.12 consente di capire subito che la codifica non può essere ottima, poiché alcune parole potrebbero essere accorciate. In generale la prestazione del codice è scadente quando n è modesto; vedremo però che le prestazioni asintotiche sono migliori. Calcoliamo la lunghezza media del codice multinomiale.

$$E[L^n] = \sum_{\boldsymbol{x}} \Pr(\boldsymbol{x})(l_{P} + l_{S}(\boldsymbol{x})) = \lceil \log_{D} \Gamma_n \rceil + \sum_{j=1}^{\Gamma_n} \sum_{\boldsymbol{x} \in T_j} \Pr(\boldsymbol{x}) \lceil \log_{D} |T_j| \rceil <$$

$$< 2 + K \log_{D}(n+1) + \sum_{j=1}^{\Gamma_n} \Pr(T_j) \log_{D} |T_j|$$

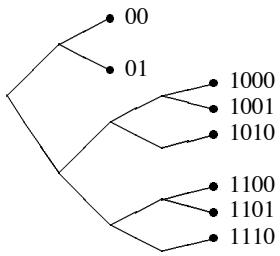


Figura 5.12 Albero del codice multinomiale relativo all'esempio 5.8.

L'ultima sommatoria dipende direttamente dal suffisso, e dimostreremo che c'è un legame con l'entropia della sorgente. Per il momento si noti che il contributo del prefisso è asintoticamente irrilevante; infatti il termine $2 + K \log_D(n+1)$ normalizzato a n tende asintoticamente a zero. Per far emergere l'entropia consideriamo una v.a. Z che rappresenta il numero d'ordine del tipo esatto associato alla realizzazione della v.a. $\boldsymbol{X}^{\boldsymbol{n}} = \boldsymbol{x}$. Z prende i suoi valori sull'insieme $1, 2, \ldots, \Gamma_n$ ed è funzione deterministica di $\boldsymbol{X}^{\boldsymbol{n}}$. Calcoliamo l'incertezza media associata a $\boldsymbol{X}^{\boldsymbol{n}}$ nota che sia Z

$$H(\boldsymbol{X}^{\boldsymbol{n}}/Z) = \sum_{j=1}^{\Gamma_n} \Pr(Z=j) H(\boldsymbol{X}^{\boldsymbol{n}}/Z=j) = \sum_{j=1}^{\Gamma_n} \Pr(T_j) H(\boldsymbol{X}^{\boldsymbol{n}}/Z=j)$$

Quando $Z = j$ abbiamo un'incertezza residua su $\boldsymbol{X}^{\boldsymbol{n}}$ che è limitata superiormente dalla cardinalità del tipo esatto $H(\boldsymbol{X}^{\boldsymbol{n}}/Z=j) \leq \log_D |T_j|$; si ricordi che l'entropia raggiunge l'uguaglianza quando la relativa d.p. è uniforme (si veda (2.15)). Nel caso in questione tutte le n-ple dello stesso tipo esatto, avendo la stessa composizione, hanno anche la medesima probabilità (nelle ipotesi di stazionarietà e assenza di memoria), e dunque la d.p. associata si presenta uniforme. In questo caso la limitazione di prima vale col segno di uguaglianza. Possiamo scrivere

$$H(\boldsymbol{X}^{\boldsymbol{n}}/Z=j) = \log_D |T_j|$$

che sostituita nell'espressione della lunghezza media porta alla seguente catena di disuguaglianze

$$\begin{aligned} E[L^n] &< 2 + K\log_D(n+1) + \sum_{j=1}^{\Gamma_n} \Pr(T_j) H(\boldsymbol{X}^{\boldsymbol{n}}/Z=j) = \\ &= 2 + K\log_D(n+1) + H(\boldsymbol{X}^{\boldsymbol{n}}/Z) \leq \\ &\leq 2 + K\log_D(n+1) + H(\boldsymbol{X}^{\boldsymbol{n}}) = \\ &= 2 + K\log_D(n+1) + nH(P) \end{aligned}$$

Normalizzando si ricava

$$\frac{E[L^n]}{n} < \frac{2}{n} + \frac{K}{n}\log_{_D}(n+1) + H(P) \tag{5.18}$$

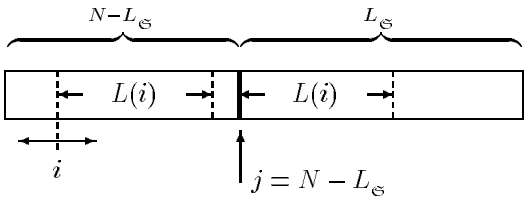
Dalla relazione si evince che il codice è *asintoticamente ottimo*, poiché il tasso tende all'entropia quando $n \to \infty$. Tutto ciò è stato realizzato *senza avere informazione alcuna sulla struttura statistica della sorgente*. Questo è dunque un esempio di codice universale, che in pratica può essere usato non solo quando la d.p. sia sconosciuta, ma anche quando il processo è non stazionario (si noti però che per l'ottenimento della (5.18) è stata supposta la stazionarietà). Il prezzo da pagare per l'universalità è nel senso della *complessità di attuazione* (il dizionario ha una cardinalità esponenziale) e nella *convergenza lenta* verso l'entropia: infatti per il codice di Shannon-Fano l'infinitesimo è $\frac{1}{n}$, mentre in questo caso è dell'ordine di $\frac{\log n}{n}$.

Oss. 5.6. L'ottimalità asintotica del codice multinomiale, pur in assenza d'informazioni sulla statistica della sorgente, si può spiegare nel modo seguente. Asintoticamente una sorgente emette "solo" sequenze tipiche in composizione (vedi sezione 3.3.2), che sono in numero di circa $2^{nH(P)}$; per individuare tra esse quella che esce effettivamente servono $\approx nH(P)$ bit, il che porta a un tasso normalizzato pari a $H(P)$. Nel caso dei codici *B-B* l'informazione sulla statistica viene usata per *discriminare a priori* le n-ple tipiche dalle altre, attribuendo parole di codice distinte *solo* alle tipiche. Nel codice multinomiale s'ignora la d.p. della sorgente e vengono invece codificate indiscriminatamente *tutte* le n-ple, a prezzo di una lunghezza media più elevata. Tuttavia la differenza (riconducibile alla specificazione del tipo, e quindi al prefisso) si annulla asintoticamente.

5.6.2 Codice di Ziv-Lempel

Il codice di Ziv-Lempel effettua una compressione dati di tipo *strutturale*, basata cioè su informazioni inerenti la struttura della concatenazione delle lettere emesse dalla sorgente. Anche in questo caso non è necessaria alcuna conoscenza a priori sulla statistica della stessa sorgente. Nella codifica si forniscono le coordinate, relative alla sequenza pregressa, necessarie per ricostruire l'informazione futura mediante replica (parziale) di quella pregressa. In altre parole si copia parte dell'informazione che è già stata generata, fornendo lunghezza e posizione della sottosequenza che deve essere replicata. Prima di descrivere l'algoritmo precisiamo alcune notazioni. Sia $\mathfrak{S} = \{s_1, s_2, \ldots, s_{_K}, \ldots\}$ una successione (infinita a destra) di lettere sull'alfabeto K-ario $\mathcal{A}$. Scriveremo $\mathfrak{S}(i,j) = (s_i, \ldots, s_j)$ per indicare la sottostringa da i a j. Prendiamo ora un registro di memoria di lunghezza N, disponiamoci in posizione $j = N - L_{_{\mathfrak{S}}}$, con $L_{_{\mathfrak{S}}} < N$, e sia $L(i)$ il più grande intero tale che

$$\mathfrak{S}(i, i + L(i) - 1) = \mathfrak{S}(j+1, j+L(i)) \tag{5.19}$$

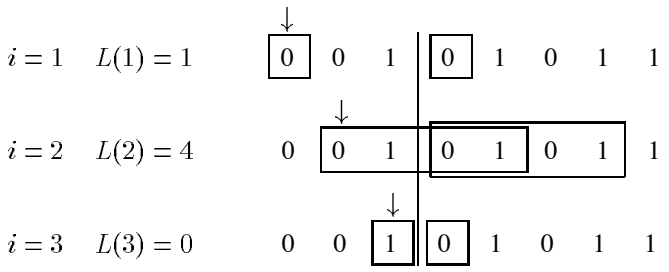


Variando i avremo diversi valori di $L(i)$ corrispondenti; consideriamo quello cui corrisponde la lunghezza massima

$$L(p) = \max_{1 \le i \le j} L(i) \tag{5.20}$$

La stringa $\mathfrak{S}(j{+}1, j{+}L(p))$ si definisce allora *estensione riproducibile* di $\mathfrak{S}(1, j)$ in $\mathfrak{S}(1, N)$, con p *puntatore della riproduzione* e $L(p)$ *lunghezza della riproduzione*. L'operazione relativa alle (5.19) e (5.20) corrisponde all'individuazione della replica di lunghezza massima ricostruibile a partire dalla posizione $j + 1$ sulla base delle informazioni pregresse, localizzate mediante il valore del puntatore p e la lunghezza $L(p)$ della riproduzione.

Esempio 5.9. Consideriamo la finestra di lunghezza 8 contenente la stringa $\mathfrak{S} = 001|01011$ con $j = 3$.



Per $i = 1$ si ha $L(1) = 1$, mentre per $i = 3$ la lunghezza massima è nulla (le due stringhe sono diverse fin dall'inizio). Il risultato migliore lo si ha per $i = 2$, dove si ricava $L(2) = 4$. $\mathfrak{S}(3{+}1, 3{+}4) = 0101$ è allora l'estensione riproducibile di $\mathfrak{S}(1, 3) = 001$ in $\mathfrak{S}$ con $p = 2$ e $L(p) = L(2) = 4$. Posizionandoci allora su $j = 3 + 1 = 4$ possiamo ricavare le successive $L(2) = 4$ lettere conoscendo la posizione $p = 2$. ◯

Analizziamo ora il vero e proprio algoritmo di codifica. Se $\mathfrak{S} = \{s_1, s_2, \ldots, s_K, \ldots\}$ è la sequenza in uscita dalla sorgente, viene fatta una segmentazione a lunghezza variabile nei messaggi $\mathfrak{S} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots\}$ dove $|\boldsymbol{m}_i| \le L_{\mathfrak{S}}$. Ciascun messaggio $\boldsymbol{m}_i$ wiene codificato in una parola di codice $\boldsymbol{w}_i$ a lunghezza costante L_C, costituita da tre parti

$$\boldsymbol{w}_i = w_{i1} w_{i2} w_{i3} \tag{5.21}$$

Il prefisso w_{i1} esprime la *posizione del puntatore*, mentre w_{i2} fornisce la *lunghezza della stringa*. w_{i3} ha invece lunghezza unitaria e contiene la prima cifra dopo la riproduzione. Questa cifra evita di bloccare l'algoritmo quando in posizione $j+1$ c'è una cifra che non era mai comparsa prima. In questo caso $L(p)=0$ (si veda il passo $i=3$ dell'esempio 5.9). La lunghezza costante della parola di codice è pari a

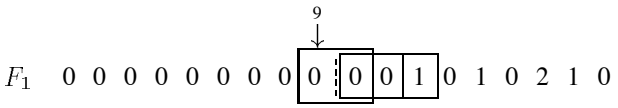
$$L_C = \lceil \log_K (N - L_{\mathfrak{S}}) \rceil + \lceil \log_K L_{\mathfrak{S}} \rceil + 1 \tag{5.22}$$

Algoritmo di Ziv-Lempel. Prima di iniziare la codifica bisogna anteporre alla successione $\mathfrak{S}$ $N - L_{\mathfrak{S}}$ zeri di inizializzazione.

1. Sia F_i il contenuto del registro di lunghezza n all'istante i. Al primo passo si ha
$$F_1 = 0^{N-L_{\mathfrak{S}}} \mathfrak{S}(1, L_{\mathfrak{S}})$$
2. Noto F_i ($i \geq 1$) sia
$$\boldsymbol{m}_i = F_i(N - L_{\mathfrak{S}} + 1, N - L_{\mathfrak{S}} + L(p_i) + 1)$$
dove le prime $L(p_i)$ cifre di $\mathfrak{S}_i$ sono l'estensione riproducibile di $F_i(1, N-L_{\mathfrak{S}})$ in $F_i(1, N-1)$, cioè $F_i(N-L_{\mathfrak{S}}+1, N-L_{\mathfrak{S}}+L(p_i))$; l'ultima cifra è la prima lettera dopo l'estensione riproducibile.
3. Se p_i è la posizione del puntatore, la parola di codice di $\mathfrak{S}_i$ è
$$\boldsymbol{w}_i = w_{i1} w_{i2} w_{i3}$$
dove w_{i1} è la rappresentazione in base K di $p_i - 1$, w_{i2} la rappresentazione in base K di $L(p_i)$ e w_{i3} è l'ultimo simbolo di $\mathfrak{S}_i$ che occupa posizione $N - L_{\mathfrak{S}} + L(p_i) + 1$ di F_i. La lunghezza della parola è data dalla (5.22).
4. per aggiornare la memoria si devono eliminare dalla stessa i primi $L(p_i)+1$ simboli, facendone entrare un numero uguale dalla sorgente
$$F_{i+1} = F_i(L(p_i) + 2, N) \mathfrak{S}(r_i + 1, r_i + L(p_i) + 1)$$
dove r_i è la posizione occupata in $\mathfrak{S}$ dall'ultimo simbolo di F_i.

La decodifica si basa sull'uso di un registro di lunghezza $N - L_{\mathfrak{S}}$, che deve essere inizialmente vuoto, e avviene impiegando le informazioni contenute nelle parole di codice per replicare in uscita l'estensione riproducibile. Vediamo un esempio completo tratto dall'articolo originale [87].

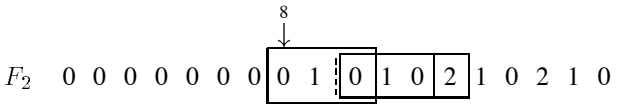
Esempio 5.10. Si abbia la seguente stringa $\mathfrak{S}$ =00101021021021202102120 0 su un alfabeto ternario, con $N = 18$ lunghezza del registro, $L_S = 9$ e $L_C = \lceil \log_3(18-9) \rceil + \lceil \log_3(9) \rceil + 1 = 5$. Aggiungiamo $N - L_S = 9$ zeri iniziali e consideriamo il contenuto della prima finestra di lunghezza 18:



Il più lungo prefisso $F_1(10, 9 + L(p_1))$ di $F_1(10, 17)$ che coincide con una sottostringa di $F_1(1, 9)$ ha lunghezza pari a 2, e lo si ottiene, p.es., con $p_1 = 9$ (in questo caso si può scegliere un qualunque p_1 compreso tra 0 e 9). Per la codifica faremo uso di una numerazione in base 3.

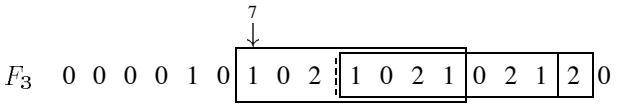
$$\begin{array}{cccc} p_1 - 1 = 8 & L(p_1) = 2 & & \\ \Downarrow & \Downarrow & & \\ w_{11} = 22 & w_{12} = 02 & w_{13} = 1 & \Longrightarrow \boldsymbol{w}_1 = 22021 \end{array}$$

$\boldsymbol{w}_1$ è la prima parola di codice, espressa in base 3, e composta dalle informazioni riguardanti la posizione e la lunghezza dell'estensione riproducibile. L'ultima cifra di $\boldsymbol{w}_1$ è quella immediatamente successiva all'estensione. Traslando la sequenza a sinistra di $L(p_1) + 1 = 3$ cifre si ottiene la nuova configurazione dalla quale ricavare la seconda parola di codice:



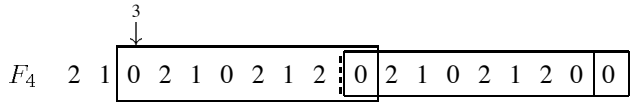
$$\begin{array}{cccc} p_2 - 1 = 7 & L(p_2) = 3 & & \\ \Downarrow & \Downarrow & & \\ w_{21} = 21 & w_{22} = 10 & w_{23} = 2 & \Longrightarrow \boldsymbol{w}_2 = 21102 \end{array}$$

In questo secondo caso la posizione del puntatore per la quale si ha la massima estensione riproducibile è determinata in modo univoco, e corrisponde a $p_2 = 8$, mentre la seconda traslazione a sinistra è pari a 4.



$$\begin{array}{cccc} p_3 - 1 = 6 & L(p_3) = 7 & & \\ \Downarrow & \Downarrow & & \\ w_{31} = 20 & w_{32} = 21 & w_{33} = 2 & \Longrightarrow \boldsymbol{w}_3 = 20212 \end{array}$$

Le ultime due traslazioni verso sinistra, rispettivamente di 8 e 9 cifre, consentono di completare la codifica



$$\begin{array}{lllll} p_4 - 1 = 2 & L(p_4) = 8 & & & \\ \Downarrow & \Downarrow & & & \\ w_{41} = 02 & w_{42} = 22 & w_{43} = 0 & \Longrightarrow \boldsymbol{w}_4 = 02220 \end{array}$$

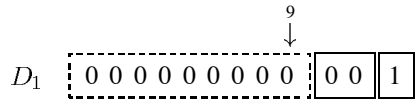
Per la decodifica abbiamo a disposizione la concatenazione delle parole di codice a lunghezza costante (pari a 5), che corrisponde alla sequenza ternaria

$$22021|21102|20212|02220$$

e un registro di lunghezza $N - L_{\mathcal{S}} = 9$ inizialmente vuoto. Scomponendo la prima parola di codice nelle sue tre sottocomponenti ricaviamo

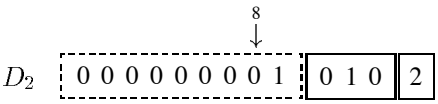
$$\begin{array}{lll} \boldsymbol{w}_1 = 22\ 02\ 1 & p_1 - 1 = (22)_3 = 8 \quad \Rightarrow \quad p_1 = 9 \\ & L(p_1) = (02)_3 = 2 \quad \Rightarrow \quad L(p_1) + 1 = 3 \end{array}$$

Agli zeri di inizializzazione bisogna allora aggiungere 3 cifre, lette a partire dalla posizione 9 del puntatore, di cui l'ultima sarà un "1".



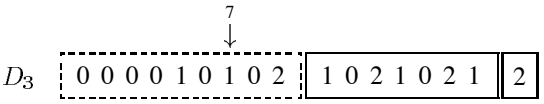
$$\begin{array}{lll} \boldsymbol{w}_2 = 21\ 10\ 2 & p_2 - 1 = (21)_3 = 7 \quad \Rightarrow \quad p_2 = 8 \\ & L(p_2) = (10)_3 = 3 \quad \Rightarrow \quad L(p_2) + 1 = 4 \end{array}$$

La decodifica di $\boldsymbol{w}_2$ comporta la lettura da posizione 8 di 3 cifre con l'aggiunta finale di un "2"



$$\begin{array}{lll} \boldsymbol{w}_3 = 20\ 21\ 2 & p_3 - 1 = (20)_3 = 6 \quad \Rightarrow \quad p_3 = 7 \\ & L(p_3) = (21)_3 = 7 \quad \Rightarrow \quad L(p_3) + 1 = 8 \end{array}$$

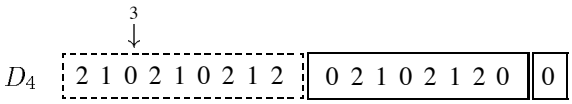
Si noti che in entrambi i casi precedenti, e nei due successivi, la lettura dell'estensione riproducibile coinvolge anche lettere appena generate. Per $\boldsymbol{w}_3$ si parte da posizione 7 e si leggono ben 7 cifre consecutive , aggiungendo alla fine un "2"



$\boldsymbol{w}_4 = 02\ 22\ 0 \qquad p_4 - 1 = (02)_3 = 2 \quad \Rightarrow \quad p_4 = 3$

$L(p_4) = (22)_3 = 8 \quad \Rightarrow \quad L(p_4) + 1 = 9$

L'ultimo passo implica l'aggiunta di 9 cifre, lette a partire dalla posizione 3, di cui l'ultima deve essere uno "0"



○

Il codice è dunque da lunghezza variable a blocco, e il suo funzionamento si basa sul fatto che le configurazioni più lunghe e più frequenti si ritrovano, del tutto o in parte, all'interno della finestra di lunghezza $N - L_{\mathfrak{S}}$; ciò consente di codificarle in modo efficiente fornendo le sole informazioni necessarie a individuare la posizione e la lunghezza della replica. Le prestazioni del codice sono scadenti quando n è modesto, come accade del resto per tutti i codici universali, ma asintoticamente il codice di Ziv-Lempel è ottimo. La dimostrazione di ciò è tuttavia complessa (si veda [87]), e in questa sede ci accontentiamo di un ragionamento euristico e molto grossolano. Osservando il funzionamento asintotico di una sorgente *SSM* sappiamo che questa emette "solo" sequenze tipiche, che sono circa $2^{nH(\mathcal{S})}$; ciascuna sequenza tipica comparirà mediamente una volta per blocco di lunghezza $2^{nH(\mathcal{S})}$; imponiamo allora $N - L_{\mathfrak{S}} = 2^{nH(\mathcal{S})}$. Dalla posizione $N - L_{\mathfrak{S}} + 1$ parte una nuova sequenza tipica, di lunghezza pari a n, che con elevata probabilità è rintracciabile (del tutto o in parte) tra le $2^{nH(\mathcal{S})}$ cifre di $N - L_{\mathfrak{S}}$. Con un calcolo molto approssimato possiamo specificare il tasso n-esimo del codice, tenuto conto che la codifica della posizione comporta circa $nH(\mathcal{S})$ bit, mentre quella della lunghezza al più $\log n$. Si ottiene così

$$R_n \approx \frac{|\boldsymbol{w}|}{n} = \frac{nH(\mathcal{S}) + \log n + 1}{n}$$

che porta a un valore asintoticamente prossimo all'entropia.

Del codice esiste anche una variante nella quale si costruisce un dizionario delle stringhe già comparse; la codifica fornisce in questo caso le informazioni necessarie per reperire la sequenza giusta all'interno del dizionario [88].

Il codice di Ziv-Lempel ha trovato largo impiego in ambito applicativo; come esempio possiamo citare il sottoprogramma *compress* in ambiente UNIX, o la famiglia degli *arc* per ambiente PC.

Capitolo 6
Codici correttori d'errore

6.1 Introduzione

Nella sezione 4.1 d'introduzione alla codifica di canale abbiamo anticipato che la strategia per rivelare ed eventualmente correggere gli errori in trasmissione è basata sull'introduzione di ridondanza strutturale e sistematica all'interno delle sequenze che codificano le lettere che escono dalla sorgente (o dal codificatore di sorgente se si è provveduto a una compressione dei dati). L'operazione viene eseguita dal *codificatore di canale*. La presenza della ridondanza si esplicita nella circostanza che tra tutte le q^n n-ple dell'alfabeto q-ario del canale ne scegliamo solo $M < q^n$ come parole di codice, costituendo così il *dizionario* $\mathfrak{D}$ *del codice*. Trasmessa $\boldsymbol{x} \in \mathfrak{D}$ all'ingresso del canale, se dall'altra parte riceviamo $\boldsymbol{y} \notin \mathfrak{D}$ si è sicuramente verificato un errore di trasmissione. Il tasso del codice, definito nella (4.4)

$$R = \frac{\log_q M}{n}$$

è la quantità d'informazione (in q-it) per lettera di parola di codice, che cresce con la cardinalità M. Un aumento eccessivo di R penalizza però la ridondanza del codice, e quindi la sua capacità di individuare e correggere gli errori. D'altra parte un valore di M troppo basso va a scapito della capacità del codice di smaltire l'informazione. Si comprende allora che la scelta di M sarà effettuata sulla base di un compromesso tra l'esigenza di smaltire molta informazione (M elevato) e quella di avere un'elevata capacità di correzione (M piccolo). Questo aspetto verrà studiato in modo sistematico, anche nei suoi risvolti asintotici, nella sezione 6.6.

Una volta fissato M, un ulteriore problema da risolvere è legato alla scelta delle parole di codice tra le q^n n-ple disponibili. Una situazione simile a quella della parte sinistra della figura 6.1 non è di certo vantaggiosa, poiché due parole di codice "vicine", secondo una qualche metrica, rischiano di essere facilmente confuse dal decodificatore che, sulla base della stessa metrica, deve effettuare la

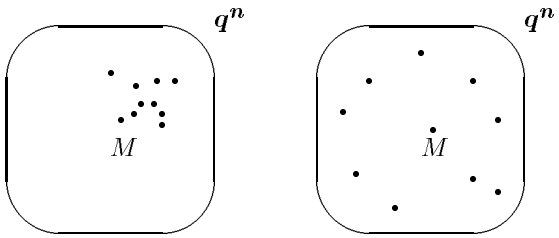


Figura 6.1 Due possibili scelte per le parole di codice.

separazione in regioni di decisione, attuando uno dei criteri di decodifica visti nella sezione 4.3. Infatti, anche se non è ancora chiarito il legame tra probabilità d'errore e distanza tra n-ple, possiamo dire grosso modo che il canale trasformerà con maggiore probabilità una parola di codice in un n-pla a essa vicina piuttosto che in una lontana; se quest'ultima è anch'essa una parola di codice sarà impossibile riconoscere l'errore. È allora opportuno che le parole di codice siano mediamente "distanti" reciprocamente. Nella costruzione del codice fatta ispirandosi alla parte destra della figura 6.1, bisognerà però tener conto della complessità delle operazioni di codifica/decodifica, fondamentali dal punto di vista delle applicazioni. Come vedremo la ricerca di un compromesso tra queste esigenze spesso contrastanti non è ancora del tutto conclusa.

D'altronde il teorema diretto di Shannon, per canali stazionari e senza memoria 4.2, aiuta poco a tal riguardo; benché si dimostri l'esistenza di successioni di codici caratterizzati da una probabilità d'errore asinototicamente nulla, a fronte di un tasso di trasmissione minore della capacità del canale (ma prossimo quanto si vuole ad essa) il teorema non è di tipo *costruttivo*, e non fornisce indicazioni concrete per realizzare tali codici, almeno se si prescinde dalla tecnica della *codifica aleatoria*, molto raffinata sul piano concettuale, ma inattuabile su quello concreto. Il carattere asintotico del procedimento impone di ottenere tutte le parole di codice con un procedimento di complessità computazionale modesta, e senza ricorrere al rozzo espediente di memorizzare sistematicamente tutte le parole, che costringerebbe a pesanti ricerche sul dizionario. È dunque necessario che le parole di codice siano legate da una qualche *struttura*.

I cinquant'anni di ricerche nell'ambito della teoria dei codici correttori che ci separano dal teorema di Shannon sono stati spesi proprio per cercare una soluzione al problema di individuare codici strutturati con garantiscano le prestazioni promesse dal teorema, ma con una elevata efficienza nelle operazioni di codifica/decodifica.

6.2 Codici algebrici

D'ora in poi ci riferiremo a strutture di tipo algebrico, anche se nei codici convolutivi ci si può allontanare in parte da quest'ipotesi.

Assegnato l'alfabeto q-ario $\mathcal{A} = \{a_1, a_2, \ldots, a_q\}$, con esso costruiremo tutte le possibili q^n stringhe di lunghezza n, che indichiamo col simbolo $\boldsymbol{q^n}$. Dalla definizione 4.1 segue che un codice blocco q-ario è un qualunque sottinsieme $\mathfrak{C} \subseteq \boldsymbol{q^n}$, mentre le stringhe $\boldsymbol{x} \in \mathfrak{C}$ sono le parole di codice. Nel seguito manterremo (quasi sempre) l'ipotesi che l'alfabeto q-ario $\mathcal{A}$ sia un *campo finito (di Galois)*, che denotiamo con $GF(q)$, dove $q = p^r$ potenza di un numero primo, e che $\boldsymbol{q^n}$ sia uno *spazio vettoriale* sul campo $GF(q)$. A beneficio del lettore facciamo una piccola digressione, passando in rapida rassegna le definizioni delle principali strutture algebriche d'interesse per la teoria dei codici. Per i dettagli rimandiamo a [18] o a [33].

6.2.1 Richiami sulle strutture algebriche

Nel seguito considereremo insiemi $\boldsymbol{S}$ di elementi e le loro corrispondenti *leggi di composizione* (interna), cioè le applicazioni $f : \boldsymbol{B} \to \boldsymbol{S}$ dove $\boldsymbol{B} \subset \boldsymbol{S} \times \boldsymbol{S}$ ($\boldsymbol{B} \neq \emptyset$).

Semigruppi. Se $\boldsymbol{S}$ è un insieme e $*$ una sua legge di composizione interna di tipo associativo, dicesi *semigruppo* la coppia $(\boldsymbol{S}, *)$, che possiamo indicare anche con $\boldsymbol{S}(*)$. Un esempio già incontrato nella sezione 3.4 è fornito dalla coppia $(\mathcal{A}^+, \circ)$, dove $\mathcal{A}^+$ è l'insieme di tutte le sequenze finite costruite con elementi di un alfabeto $\mathcal{A}$, e $\circ$ è la legge di *concatenazione* tra sequenze. Un semigruppo dotato di *unità* o *elemento neutro* dicesi *monoide*. Ricordiamo che $\boldsymbol{u}$ è neutro per la legge $*$ se $\forall \boldsymbol{x} \in \boldsymbol{S}$ si ha $\boldsymbol{x} * \boldsymbol{u} = \boldsymbol{u} * \boldsymbol{x} = \boldsymbol{x}$.

Gruppi. Dicesi *gruppo* $\boldsymbol{G}$ ogni monoide $(\boldsymbol{G}, \cdot)$ per il quale, se $\mathbf{1}$ è l'unità del monoide, per ogni $\boldsymbol{x} \in \boldsymbol{G}$ esiste un unico reciproco $\boldsymbol{x^{-1}} \in \boldsymbol{G}$ tale che $\boldsymbol{x} \cdot \boldsymbol{x^{-1}} = \boldsymbol{x^{-1}} \cdot \boldsymbol{x} = \mathbf{1}$. Se la legge di composizione è commutativa, il gruppo si dice *abeliano*. Naturalmente $\mathbf{1}$ è inverso di sé stesso. Al posto della notazione $\cdot$ (gruppo moltiplicativo) si può usare anche la $+$, nel qual caso si parla di *gruppo additivo*, del suo *zero* e dell'*opposto* di un qualunque elemento.

Se $\boldsymbol{H}$ è un sottinsieme non vuoto di $\boldsymbol{G}$, diremo che esso è un suo *sottogruppo* se e solo se, per ogni coppia $\boldsymbol{a}$, $\boldsymbol{b}$ di elementi di $\boldsymbol{H}$, anche $\boldsymbol{ab^{-1}} \in \boldsymbol{H}$.

Laterali di un sottogruppo. Sia G un gruppo commutativo (additivo) e H un suo sottogruppo. Diciamo che due elementi x e y di G sono equivalenti se e solo se $x - y \in H$, dove con $-y$ indichiamo l'opposto di y. Detta $[x]$ la classe di equivalenza di x, per un arbitrario $y \in [x]$ accade che $x - y = h'$, con h' elemento di H; cioè $x = h' + y$ oppure $y = x + h$ con $h = -h'$. Dunque tutti gli elementi di $[x]$ sono del tipo $y = x + h$, e ciò suggerisce di indicare la classe $[x]$ mediante il simbolo $x + H$. Le classi di equivalenza $[x] = x + H$ sono dette *laterali del sottogruppo*.

Gruppi ciclici. Sia a un elemento di un gruppo moltiplicativo G e si considerino le sue potenze $a^1, a^2, \ldots a^k, \ldots$. Se il gruppo è finito esistono a^h e a^k tali che $a^h = a^k$, cioè $a^{h-k} = 1$. Il più piccolo intero tale che $a^n = 1$ si dice *ordine* di a . Consideriamo ora il *sottogruppo generato* da a, che si indica con la notazione $\langle a \rangle$, e che corrisponde all'intersezione di tutti i sottogruppi di G che contengono a. Poiché tutte le potenze di a sono contenute in $\langle a \rangle$ e inoltre costituiscono esse stesse un sottogruppo, esse coincidono con $\langle a \rangle$, che viene definito *sottogruppo ciclico*. Un gruppo G che coincida con uno dei suoi sottogruppi ciclici si dice *ciclico*.

Anelli. Un anello è un insieme non vuoto R in cui sono definite due leggi di composizione interna, $+$ e $\cdot$, tali che la struttura $(R, +)$ sia un gruppo commutativo e la $(R, \cdot)$ un semigruppo. Presi inoltre tre elementi qualunque a, b, c di R, valgono le seguenti leggi distributive: $a(b+c) = ab+ac$ e $(b+c)a = ba+ca$. L'anello è *con unità* se la sua struttura moltiplicativa possiede un'unità, ed è *commutativo* se la stessa struttura è commutativa. Un sottinsieme T è un *sottoanello* di R se, dati due arbitrari elementi a, b di T, anche gli elementi $a - b$ e $a \cdot b$ appartengono a T.

Un sottoanello I di un anello R si dice *ideale* di R se per ogni $a \in I$ e $r \in R$ si ha $r \cdot a \in I$ e $a \cdot r \in I$.

Corpi e campi. Si è detto che in un anello $R(+, \cdot)$ la struttura additiva $R(+)$ è un gruppo, mentre quella moltiplicativa $R(\cdot)$ è un semigruppo. Anche immaginando che $R(\cdot)$ sia un monoide non si può sperare che diventi un gruppo, poichè l'elemento neutro di $R(+)$, cioè lo zero, non è invertibile. Può invece accadere che sia un gruppo l'insieme $(R - \{0\})(\cdot)$. Dicesi allora *corpo* un anello i cui elementi non nulli formano un gruppo moltiplicativo. Un corpo commutativo si dice *campo*. Un campo con un numero finito di elementi si chiama *campo finito di Galois*, e lo si indica con la notazione $GF(q)$, con q ordine del campo .

Spazi vettoriali. Siamo assegnati un gruppo additivo G, i cui elementi denoteremo con $x, y, z, \ldots$, e un campo F, con elementi $\alpha, \beta, \gamma, \ldots$. Sia inoltre Φ una *legge di composizione esterna*, cioè un'applicazione $\Phi : B \to G$,

dove $\boldsymbol{B} \subset F \times \boldsymbol{G}$ ($\boldsymbol{B} \neq \emptyset$). Denotiamo con $\alpha\boldsymbol{x}$ l'elemento $\alpha\Phi\boldsymbol{x}$. Il gruppo $\boldsymbol{G}$ si dice allora *spazio vettoriale* su F se la Φ gode delle seguenti proprietà: $\alpha(\boldsymbol{x}+\boldsymbol{y}) = \alpha\boldsymbol{x} + \alpha\boldsymbol{y}$, $(\alpha+\beta)\boldsymbol{x} = \alpha\boldsymbol{x} + \beta\boldsymbol{y}$, $(\alpha \cdot \beta)\boldsymbol{x} = \alpha(\beta\boldsymbol{x})$.

Nella sezione 6.4.1 forniremo qualche informazione anche sulla struttura di un campo finito, sulla rappresentazione dei suoi elementi e sull'impatto che tale rappresentazione ha nell'ambito delle operazioni di codifica/decodifica.

6.2.2 Spazio di Hamming

Si consideri il più semplice modello di canale, quello simmetrico, binario stazionario e senza memoria, con una probabilità di transizione pari a ϵ (analizzato nel paragrafo 4.2.5). Se $\boldsymbol{x} = x_1x_2\ldots x_n \in \mathfrak{D}$ è un vettore n-dimensionale d'ingresso, il rumore agisce su ciascuna componente x_i in modo indipendente dalle altre: con probabilità $\epsilon > 0$ x_i diventa $\bar{x_i}$, cioè 0 diventa 1 o viceversa. Se il canale è q-ario, con probabilità ϵ ciascun x_i diventa uno dei $q-1$ simboli rimanenti. Si può dunque pensare che l'effetto del rumore sia quello di sovrapporre a ogni componente x_i un *segnale di rumore* e_i, con $e_i = 0$ se il rumore non sortisce alcun effetto sulla i-esima componente. Il vettore $\boldsymbol{e}$ così determinato costituisce l'*n-pla d'errore*. Possiamo dunque scrivere

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \tag{6.1}$$

con $\boldsymbol{y}$ vettore d'uscita dal canale. Se $\boldsymbol{e} = \boldsymbol{0}$ non si sono verificati errori sul blocco di lunghezza n, e $\boldsymbol{y} = \boldsymbol{x}$. Viceversa se $e_i \neq 0$ allora si è verificato un errore in posizione i. Diamo alcune definizioni.

Def. 6.1. *Assegnati i vettori $\boldsymbol{x} = x_1x_2\ldots x_n$ e $\boldsymbol{y} = y_1y_2\ldots y_n$ si definisce distanza di Hamming $d_H(\boldsymbol{x},\boldsymbol{y})$ tra $\boldsymbol{x}$ e $\boldsymbol{y}$ il numero di posizioni per le quali $x_i \neq y_i$*

$$d_H(\boldsymbol{x},\boldsymbol{y}) = |\{i : x_i \neq y_i\}| \tag{6.2}$$

Così $d_H(01101, 10101) = 2$ e $d_H(213011, 123013) = 3$.

Def. 6.2. *Assegnato un vettore $\boldsymbol{x} = x_1x_2\ldots x_n$ si definisce peso del vettore la sua distanza di Hamming dall'n-pla nulla (il numero delle componenti diverse da 0)*

$$wt(\boldsymbol{x}) = d_H(\boldsymbol{x}, \boldsymbol{0}) \tag{6.3}$$

Per esempio $wt(01101) = 3$ e $wt(213011) = 5$. Si noti che

$$d_H(\boldsymbol{x},\boldsymbol{y}) = wt(\boldsymbol{x}-\boldsymbol{y}) \tag{6.4}$$

e che esistono $\binom{n}{i}(q-1)^i$ n-ple q-arie di peso i.

La distanza di Hamming è una *metrica*, come si verifica facilmente

$$\begin{aligned} &i) && d_{_H}(\boldsymbol{x},\boldsymbol{y}) = 0 \quad \text{sse} \quad \boldsymbol{x}=\boldsymbol{y} \\ &ii) && d_{_H}(\boldsymbol{x},\boldsymbol{y}) = d_{_H}(\boldsymbol{y},\boldsymbol{x}) \\ &iii) && d_{_H}(\boldsymbol{x},\boldsymbol{y}) + d_{_H}(\boldsymbol{y},\boldsymbol{z}) \geq d_{_H}(\boldsymbol{x},\boldsymbol{z}) \end{aligned}$$

D'ora in poi considereremo lo spazio vettoriale $\boldsymbol{q^n}$ associato alla metrica di Hamming, cioè lo *spazio metrico di Hamming*.

Fissato un elemento $\boldsymbol{x}$ dello spazio, possiamo costruire la *sfera* centrata in $\boldsymbol{x}$ e di raggio ρ

$$S_\rho(\boldsymbol{x}) \triangleq \{\boldsymbol{y} : d_H(\boldsymbol{x},\boldsymbol{y}) \leq \rho\} \tag{6.5}$$

il cui *volume* è dato dal numero di *n*-ple che la compongono:

$$\text{Vol}[S_\rho(\boldsymbol{x})] \triangleq |S_\rho(\boldsymbol{x})|$$

Per calcolare il volume bisogna contare il numero di n-ple che stanno sui gusci a distanza i e sommare con $i : 0 \leq i \leq \rho$. Il volume di ciascun guscio è pari a

$$\text{Vol}[g_i(\boldsymbol{c})] = \binom{n}{i}(q-1)^i \tag{6.6}$$

dove $\binom{n}{i}$ è il numero di possibili posizioni nelle quali cambiare il simbolo e $q-1$ è il numero di simboli da mettere al posto del simbolo cambiato. Il volume della sfera diventa dunque

$$\text{Vol}[S_\rho(\boldsymbol{c})] = \sum_{i=0}^{\rho} \binom{n}{i}(q-1)^i \tag{6.7}$$

Una volta trasmesso $\boldsymbol{x}$ si riceve $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ e la distanza (di Hamming) tra i due vettori $d_H(\boldsymbol{x},\boldsymbol{y})$, pari al peso $wt(\boldsymbol{e})$ dell'ennupla d'errore, dà una misura di quanto il canale abbia effettivamente interferito sulla trasmissione di $\boldsymbol{x}$. Se $wt(e)$ è grande, c'è il rischio che ci si sia avvicinati troppo a un'altra parola di codice $\boldsymbol{z}$, e ciò potrebbe indurre in errore il decodificatore, che potrebbe attribuire $\boldsymbol{y}$ a $\boldsymbol{z}$. Per decodificare usiamo dunque il seguente criterio

Criterio di decodifica a distanza minima. Assegnato il dizionario $\mathfrak{D}$ e ricevuto $\boldsymbol{y} \in \boldsymbol{q^n}$ si cerca, tra tutti gli elementi $\boldsymbol{x} \in \mathfrak{D}$, quello per cui $d_{_H}(\boldsymbol{x},\boldsymbol{y})$ è minima; si decodifica $\boldsymbol{y}$ con $\boldsymbol{x}$.

Vale allora la seguente

Proposizione 6.1. *Il criterio di decodifica a distanza minima corrisponde a un criterio di massima verosimiglianza*.

Dim. Supponiamo di usare uno schema di decodifica a massima verosimiglianza per le n-ple che escono dal canale *CSB*(ϵ). Ricevuto $\boldsymbol{y}$ il decodificatore fornisce una stima

$$\widehat{\boldsymbol{x}} = \boldsymbol{x} \quad \text{se} \quad p(\boldsymbol{y}/\boldsymbol{x}) = \max_{1 \le i \le M} p(\boldsymbol{y}/\boldsymbol{x}_i)$$

La $p(\boldsymbol{y}/\boldsymbol{x})$ è data dall'espressione

$$\begin{aligned} p(\boldsymbol{y}/\boldsymbol{x}) =& p(\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}/\boldsymbol{x}) = p(\boldsymbol{e}/\boldsymbol{x}) = \\ =& p(\boldsymbol{e}) = \epsilon^{d_H(\boldsymbol{x},\boldsymbol{y})}(1-\epsilon)^{n-d_H(\boldsymbol{x},\boldsymbol{y})} = (1-\epsilon)^n \left(\frac{\epsilon}{1-\epsilon}\right)^{d_H(\boldsymbol{x},\boldsymbol{y})} \end{aligned} \tag{6.8}$$

Fissato $\boldsymbol{y}$ si tratta dunque di massimare tale quantità scegliendo un $\boldsymbol{x}$ opportuno . Poiché $(1-\epsilon)^n$ non dipende da $\boldsymbol{x}$ bisogna massimare $(\frac{\epsilon}{1-\epsilon})^{d_H(\boldsymbol{x},\boldsymbol{y})}$. Se $\epsilon < 1/2$ si ha anche $\epsilon/(1-\epsilon) < 1$ e la funzione è decrescente con $d_H(\boldsymbol{x},\boldsymbol{y})$. Ciò equivale a cercare il minimo dell'esponente:

$$\widehat{\boldsymbol{x}} = \boldsymbol{x} \quad \text{se} \quad d_H(\boldsymbol{x},\boldsymbol{y}) = \min_{1 \le i \le M} d_H(\boldsymbol{x}_i,\boldsymbol{y})$$

il che corrisponde all'impiego del criterio a distanza minima. Le cose non cambiano passando a un canale simmetrico q-ario, con l'ipotesi $0 < \epsilon < (q-1)/q$

$$p(\boldsymbol{y}/\boldsymbol{x}) = \left(\frac{\epsilon}{q-1}\right)^{d_H(\boldsymbol{x},\boldsymbol{y})} (1-\epsilon)^{n-d_H(\boldsymbol{x},\boldsymbol{y})} = (1-\epsilon)^n \left(\frac{\epsilon}{(q-1)(1-\epsilon)}\right)^{d_H(\boldsymbol{x},\boldsymbol{y})}$$

Per le ipotesi fatte si ha $\frac{\epsilon}{(q-1)(1-\epsilon)} < 1$ ottenendo nuovamente la corrispondenza tra criterio di massima verosimiglianza e criterio di distanza minima. □

Oss. *6.1*. Dalla (6.8) si evince che la probabilità dell'ennupla d'errore è decrescente col peso della stessa, e dunque le n-ple d'errore più leggere sono le più probabili.

La decodifica a distanza minima crea una partizione in $\boldsymbol{q^n}$ che corrisponde alla mappatura delle regioni di decodifica di figura 4.5.

Si consideri ora la seguente

Def. 6.3. *Dato un codice $\mathfrak{C}$ e il suo dizionario, si definisce distanza minima del codice la minima distanza tra tutte le possibili coppie di n-ple appartenenti al dizionario*

$$d_{min} = \min_{\substack{\boldsymbol{x},\boldsymbol{y} \in \mathfrak{C} \\ \boldsymbol{x} \neq \boldsymbol{y}}} d_H(\boldsymbol{x},\boldsymbol{y})$$

Si costruisca ora, secondo la definizione (6.5), una sfera centrata in $\boldsymbol{x}$ per ogni $\boldsymbol{x}$ parola di codice. Se d_{min} è la distanza minima del codice, il massimo raggio t delle sfere che consente di mantenerle tutte disgiunte è pari a

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \tag{6.9}$$

Se $wt(\boldsymbol{e}) \leq t$ la decodifica a distanza minima porta in ogni caso a una decodifica corretta, poiché $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ cade all'interno della sfera centrata in $\boldsymbol{x}$, e questa è la prestazione minima garantita. Qualunque aumento del raggio al di sopra del valore specificato dalla (6.9) comporta la sovrapposizione parziale di almeno una coppia di sfere, precisamente di quelle che determinano la d_{min}, impedendo la decodifica corretta di tutte le n-ple che cadono nell'intersezione.

Se dunque un codice ha distanza minima d_{min}, esso è in grado di correggere tutte le configurazioni fino a t errori, con t dato dalla (6.9), più eventualmente qualche altra configurazione relativa a n-ple che cadono all'esterno dell'unione delle sfere. Se viceversa si impone al codice di correggere tutte le configurazioni fino a t errori, allora bisogna disporre di sfere di raggio pari almeno a t, e per la distanza minima tra esse deve valere la relazione

$$d_{min} \geq 2t + 1 \tag{6.10}$$

dove il termine 1 garantisce la disgiunzione tra le sfere. Risulta allora ragionevole definire come *capacità di correzione* del codice la quantità

$$\lambda = \frac{d_{min}}{n} \tag{6.11}$$

Naturalmente il codice può essere usato anche come semplice *rivelatore d'errore*, nel qual caso esso è in grado di rivelare tutte le configurazioni di errore fino a $d_{min} - 1$.

Oss. 6.2. Un codice correttore con $d_{min} = 2t$ ha la stessa capacità di correzione di un codice con $d_{min} = 2t - 1$. Ciò non vale però per la capacità di rivelazione.

Esempio 6.1 (Codice a ripetizione). Si consideri un codice binario con $n = 3$ e parole di codice 000 e 111. Si ha $d_{min} = 3$ e dunque il codice corregge tutte le configurazioni fino a 1 errore. Usando un criterio di decodifica a distanza minima si crea una partizione nelle due regioni di decodifica $\mathcal{R}_{000} = \{000, 001, 010, 100\}, \mathcal{R}_{111} = \{111, 011, 101, 110\}$.

$\boldsymbol{e}$	$wt(\boldsymbol{e})$	$P_e(\boldsymbol{x_i})$	$P_e(0.1)$	
000	0	$(1-\epsilon)^3$	0.729	$P_C = 0.972$
001	1	$\epsilon(1-\epsilon)^2$	0.081	
010				
100				
011	2	$\epsilon^2(1-\epsilon)$	0.009	$P_e = 0.028$
101				
110				
111	3	ϵ^3	0.001	

Nella tabella sono riportate le n-ple d'errore e le rispettive probabilità associate al caso $\epsilon = 0.1$; nell'ultima colonna vengono riportate le probabilità di corretta decodifica P_C e la probabilità d'errore P_e. Nel caso dei codici a ripetizione la decodifica a distanza minima diventa in pratica una decodifica *a maggioranza*, nella quale fissata $d_{min} = 2t + 1$ (per un codice t-correttore) si attua la seguente decisione:

$$\begin{aligned} &\text{se} \quad 0 \leq wt(\boldsymbol{y}) \leq t \quad &&\text{si decodifica } \hat{\boldsymbol{x}} = 000 \\ &\text{se} \quad t+1 \leq wt(\boldsymbol{y}) \leq 2t+1 \quad &&\text{si decodifica } \hat{\boldsymbol{x}} = 111 \end{aligned}$$

In generale la probabilità d'errore per un codice a ripetizione t-correttore si calcola tenendo conto dell'espressione (6.8)

$$P_e = Pr(wt(\boldsymbol{e}) \geq t+1) = \sum_{i=t+1}^{2t+1} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i} \tag{6.12}$$

Un codice a ripetizione corregge dunque tutte e sole le $\sum_{i=0}^{t} \binom{n}{i}$ configurazioni di errore di peso $i \leq t$. Le sue prestazioni asintotiche sono scadenti dal punto di vista del tasso, poiché ciascuna n-pla contiene un solo bit d'informazione

$$R_R = \frac{k}{n} = \frac{1}{n} \longrightarrow 0 \tag{6.13}$$

mentre la capacità di correzione è la massima possibile

$$\lambda_R = \frac{d_{min}}{n} = \frac{n}{n} = 1 \tag{6.14}$$

◯

Si noti che può accadere che l'unione di tutte le sfere di raggio t esaurisca $\boldsymbol{q^n}$; questo succede quando

$$\text{Vol}[S_t] \cdot \text{Num}[S_t] = q^n \tag{6.15}$$

dove Vol$[S_t]$ è il volume di ciascuna sfera e Num$[S_t]$ è il numero totale di sfere. In tal caso il codice si dice *perfetto*. Tenuto conto della (4.36) la condizione di perfezione si può esprimere nel seguente modo

$$M\left[\sum_{i=0}^{t}\binom{n}{i}(q-1)^i\right]=q^n \tag{6.16}$$

Nell'esempio visto sopra con $n=3$ si hanno in tutto 8 n-ple e due parole di codice; ciascuna regione di decodifica contiene, oltre alla parola di codice, tutte le n-ple a distanza unitaria e nessun'altra, costituendo così una *sfera di Hamming* di raggio unitario centrata nella corrispondente parola di codice. Il prodotto del numero di sfere per il volume esaurisce lo spazio e dunque *il codice a ripetizione è perfetto*. Ciò vale in generale per qualunque codice a ripetizione con $d_{min}=2t+1=n$, poiché sussiste la relazione

$$\sum_{i=0}^{n}\binom{n}{i}=2^n \tag{6.17}$$

Il problema dell'individuazione delle condizioni che garantiscono la validità della (6.15), e quindi della ricerca di codici perfetti, sarà analizzato nei dettagli nell'ambito della sezione 6.3.6.

6.3 Codici lineari

6.3.1 Codifica e matrice generatrice

Come riferito nella definizione 4.1 un codice di canale è un qualunque sottinsieme $\mathfrak{C}\subseteq \boldsymbol{q^n}$. La cardinalità di $\mathfrak{C}$ determina il tasso del codice, cioè la sua capacità di smaltimento dell'informazione, mentre la modalità di selezione delle $M=|\mathfrak{C}|$ parole di codice influisce sulla capacità di correzione (o di rivelazione) degli errori. Le parole dovranno essere scelte in modo da essere mutuamente distanti, per rendere massima la d_{min}, e possibilmente strutturate, per consentire operazioni di codifica/decodifica che siano semplici ed efficienti. Questa seconda richiesta limita fortemente la libertà d'azione, e ha di fatto costituito un grosso ostacolo all'individuazione dei codici ottimi promessi dal teorema di Shannon. D'altra parte non è ragionevole ipotizzare l'impiego pratico di codici non strutturati; per convincersene si pensi a una lunghezza del blocco pari a $n=100$ con un alfabeto binario. Se imponiamo un tasso pari a $1/2$ avremo a disposizione $2^{50}\approx 1.13\ 10^{15}$ parole di codice, che non è immaginabile trattare senza una struttura che eviti la loro memorizzazione.

Per $\mathfrak{C}$ si può usare la stessa struttura di *spazio vettoriale* di cui è dotato $\boldsymbol{q^n}$; ciò comporta che *la somma di due parole di codice è ancora una parola di codice*.

Def. 6.4. *Un codice lineare di lunghezza n e dimensione k è un sottospazio $V^{(k)}$ (di dimensione q^k) dello spazio $\boldsymbol{q^n} \equiv V^{(n)}$. Esso si denota con il simbolo $C(n,k)$ o anche $C(n,k,d_{min})$.*

Per descrivere un codice lineare si deve fornire una *base*, costituita da k vettori linearmente indipendenti di lunghezza n, formata dalle righe di una matrice $\boldsymbol{G}$ di tipo $k \times n$ denominata *matrice generatrice del codice*

$$\boldsymbol{G} = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ & \vdots & & \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{pmatrix} \tag{6.18}$$

Le parole di codice si ottengono come *combinazione lineare delle righe*, cioè degli elementi della base. Permutando le righe si ottiene un codice *equivalente* .

Esempio 6.2. Si consideri il codice $C(5,1)$ con matrice generatrice

$$\boldsymbol{G_1} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Le parole di codice sono 11111 e 00000. Si tratta di un *codice a ripetizione*, con $d_{min} = 5$, che può correggere tutte le configurazioni fino a 2 errori. ◯

Esempio 6.3. Il codice $C(5,3)$ con matrice generatrice

$$\boldsymbol{G_2} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

è formato dalle seguenti parole di codice 11100, 00110, 11111, 11010, 00011, $11001, 00101$, che si ottengono come combinazione lineare delle tre righe; si noti che la presenza della riga 11111 implica che accanto ad ogni vettore di peso j esista anche un vettore di peso $n-j$. La distanza minima è pari a 2 e il codice può rivelare errori di peso unitario, ma non è in grado di correggere *tutte* le configurazioni d'errore di peso 1, poiché per far questo è necessario disporre almeno di una $d_{min} = 3$. ◯

Esempio 6.4. Il codice $C(7,4)$ con matrice generatrice

$$\boldsymbol{G_3} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

è espresso nella sua *forma canonica* $\boldsymbol{G} = (I_k \ A)$, con I_k matrice identica. Un codice $C(n,k)$ espresso mediante una matrice generatrice in forma canonica si dice *sistematico* ◯

Poiché in un codice lineare di dimensione k ci sono $M = q^k$ parole di codice, il suo tasso è pari a

$$R = \frac{\log_q q^k}{n} = \frac{k}{n} \tag{6.19}$$

e dunque delle n lettere per parola di codice k sono d'informazione e $n-k$ sono di ridondanza (o di *controllo*). La *codifica* avviene associando a ciscuna k-pla $\boldsymbol{u} = u_1 u_2 \dots u_k$ di lettere d'informazione una parola di codice $\boldsymbol{x} = x_1 x_2 \dots x_n$ di lunghezza n. Ciò può avvenire direttamente mediante *prodotto riga per colonna* tra il vettore $\boldsymbol{u}$ e la matrice $\boldsymbol{G}$

$$\boldsymbol{x} = \boldsymbol{u} \cdot \boldsymbol{G} \tag{6.20}$$

che corrisponde a una trasformazione $\boldsymbol{G} : V^{(k)} \to V^{(n)}$ da uno spazio di dimensione k a uno di dimensione n; la dimensione dell'immagine è ancora k.

Esempio 6.5. Riprendiamo le matrici $\boldsymbol{G_1}$, $\boldsymbol{G_2}$ e $\boldsymbol{G_3}$ dell'esempio precedente e troviamo la struttura delle parole di codice

$$(u_1) \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} u_1 & u_1 & u_1 & u_1 & u_1 \end{pmatrix}$$

$$\begin{pmatrix} u_1 & u_2 & u_3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} =$$
$$\begin{pmatrix} u_1+u_3 & u_1+u_3 & u_1+u_2+u_3 & u_2+u_3 & u_3 \end{pmatrix}$$

$$\begin{pmatrix} u_1 & u_2 & u_3 & u_4 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} =$$
$$\begin{pmatrix} u_1 & u_2 & u_3 & u_4 & u_2+u_3+u_4 & u_1+u_3+u_4 & u_1+u_2+u_4 \end{pmatrix}$$

Nell'ultimo caso le prime $k = 4$ cifre sono replicate dal messaggio, e sono dunque le cifre d'informazione; le restanti $n - k = 3$ sono invece cifre di controllo. Ciò accade perché la matrice è in forma canonica. ◯

6.3.2 Decodifica e matrice di controllo

Accanto alla descrizione di un codice lineare basata sullo spazio $V^{(k)}$ e sulla matrice generatrice $\boldsymbol{G}$, ci si può riferire anche al *sottospazio ortogonale*

$$\begin{aligned} C^{\perp}(n,k) &= \{\boldsymbol{y} : \boldsymbol{y} \perp \boldsymbol{x},\ \forall \boldsymbol{x} \in \mathfrak{C}\} = \\ &= \{\boldsymbol{y} : \boldsymbol{y} \cdot \boldsymbol{x} = \boldsymbol{0},\ \forall \boldsymbol{x} \in \mathfrak{C}\} \end{aligned}$$

di dimensione $n-k$, che è ortogonale a $C(n,k)$; esso rappresenta un codice $C(n,n-k)$ detto anche *codice duale*. Per ciascun vettore $\boldsymbol{x} \in \mathfrak{C}$ vale la seguente relazione di *ortogonalità* (in $GF(q)$)

$$x_1 y_1 + x_2 y_2 + \ldots + x_n y_n \equiv 0 \tag{6.21}$$

che costituisce *un'equazione di controllo*. Di queste equazioni ce ne sono q^{n-k} per ogni parola di codice. Possiamo allora prendere una base di $C^{\perp}(n,k) = C(n,n-k)$, costituita da $n-k$ vettori linearmente indipendenti, e costruire con essa la seguente *matrice di controllo*

$$\boldsymbol{H} = \begin{pmatrix} h_{11} & h_{12} & \ldots & h_{1n} \\ h_{21} & h_{22} & \ldots & h_{2n} \\ & \vdots & & \\ h_{(n-k)1} & h_{(n-k)2} & \ldots & h_{(n-k)n} \end{pmatrix} \tag{6.22}$$

In questo modo ogni parola del codice sarà ortogonale a ogni riga di $\boldsymbol{H}$, che è la matrice generatrice del codice duale $C^{\perp}(n,k)$. Vale allora la seguente doppia implicazione (con $\boldsymbol{x}^T$ intendiamo il trasposto di $\boldsymbol{x}$ e con $\mathbf{0}$ il vettore nullo)

$$\boldsymbol{x} \in \mathfrak{C} \quad \Longleftrightarrow \quad \boldsymbol{H} \cdot \boldsymbol{x}^T = \mathbf{0} \tag{6.23}$$

Per assegnare un codice lineare si possono dunque fornire tanto la matrice $\boldsymbol{G}$ che la $\boldsymbol{H}$. Se la $\boldsymbol{G}$ è in forma canonica si può ricavare immediatamente la $\boldsymbol{H}$

$$\boldsymbol{G} = \begin{pmatrix} I_k & A \end{pmatrix} \qquad \boldsymbol{H} = \begin{pmatrix} -A^T & I_{n-k} \end{pmatrix} \tag{6.24}$$

e si verifica che $\boldsymbol{H}\,\boldsymbol{G^T} = \mathbf{0}$.

Sia dato ora $\boldsymbol{x} \in \mathfrak{C}$; poiché $\boldsymbol{x}$ è una parola di codice vale la relazione $\boldsymbol{H} \cdot \boldsymbol{x}^T = \mathbf{0}$. Se mettiamo in evidenza la struttura per colonne della matrice di controllo

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{h_1} & \boldsymbol{h_2} & \ldots & \boldsymbol{h_n} \end{pmatrix} \tag{6.25}$$

dove $\boldsymbol{h_i}$ è la i-esima colonna di $\boldsymbol{H}$, il prodotto $\boldsymbol{H} \cdot \boldsymbol{x}^T$ è una somma di alcune colonne di $\boldsymbol{H}$, in numero pari la peso del vettore $\boldsymbol{x}$. La condizione da soddisfare per l'appartenenza al codice si può allora esprimere anche nel modo seguente

$$\boldsymbol{H} \cdot \boldsymbol{x}^T = \sum_{i=1}^{n} x_i \boldsymbol{h_i} = \mathbf{0} \tag{6.26}$$

Se dunque il peso di una parola di codice $\boldsymbol{x} \in \mathfrak{C}$ è $wt(\boldsymbol{x})$, nella matrice $\boldsymbol{H}$ ci sono $wt(\boldsymbol{x})$ colonne la cui somma è nulla. Viceversa, per ogni combinazione lineare nulla di w colonne di $\boldsymbol{H}$ esiste nel codice $\boldsymbol{x} : wt(\boldsymbol{x}) = w$. Questo fatto

consente di sfruttare la $\boldsymbol{H}$ per determinare la distanza minima del codice; dalle (6.4) e (6.3) si ricava infatti

$$d_{min} = \min_{\substack{\boldsymbol{x},\boldsymbol{y} \in \mathfrak{C} \\ \boldsymbol{x} \neq \boldsymbol{y}}} d_{H}(\boldsymbol{x},\boldsymbol{y}) = \min_{\substack{\boldsymbol{x},\boldsymbol{y} \in \mathfrak{C} \\ \boldsymbol{x} \neq \boldsymbol{y}}} wt(\boldsymbol{x}-\boldsymbol{y}) = \min_{\substack{\boldsymbol{z} \in \mathfrak{C} \\ \boldsymbol{z} \neq \boldsymbol{0}}} wt(\boldsymbol{z})$$

poichè per un codice lineare $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{y}$ è sempre una parola di codice. Dunque in un codice lineare *la distanza minima corrisponde al peso minimo delle parole del codice*.

Esempio 6.6. Si consideri la matrice di controllo $\boldsymbol{H_3}$ associata al codice $C(7,4)$ dell'esempio precedente, la cui matrice generatrice è la $\boldsymbol{G_3}$. Usando la (6.24) si ricava

$$\boldsymbol{H_3} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Il peso minimo delle parole di codice non può essere 1, altrimenti dovrebbe esistere una colonna nulla $\boldsymbol{h_i} = \boldsymbol{0}$ per qualche i, con i posizione di un '1' all'interno della parola di codice. D'altra parte non può neanche essere 2, poiché in tal caso dovrebbero esserci in $\boldsymbol{H_3}$ (nel caso binario) due colonne identiche $\boldsymbol{h_i}, \boldsymbol{h_j}$ tali che $\boldsymbol{h_i} + \boldsymbol{h_j} = \boldsymbol{0}$, con i e j posizioni dei due 1 nella parola di codice. Poiché ci sono tre colonne linearmente indipendenti (si prendano ad esempio le prime tre), il peso minimo delle parole del codice (e di conseguenza la sua distanza minima) è pari a tre. In generale, per $m = 2, 3, 4, \ldots$ possiamo costruire una matrice $\boldsymbol{H}$ di controllo con $n = 2^m - 1$ colonne m-ple distinte eccetto la nulla. In tal caso $m = n - k$ e $k = 2^m - 1 - m$. Così facendo si ottiene un codice di *Hamming* del tipo $C(2^m - 1, 2^m - 1 - m)$ che permette di correggere 1 errore, essendo $d_{min} = 3$ (si veda la sezione 6.3.3). ◯

Poiché il canale è affetto da rumore, trasmessa $\boldsymbol{x}$ in uscita dal canale non si ottiene necessariamente la stessa parola di codice, bensì un'ennupla arbitraria del tipo $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$, con $\boldsymbol{y} \in V^n$ ed $\boldsymbol{e}$ n-pla d'errore.

Def. 6.5. *Assegnato $\boldsymbol{y} \in V^n$ si dice sindrome di $\boldsymbol{y}$ il vettore*

$$\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{H} \cdot \boldsymbol{y}^T \tag{6.27}$$

La definizione coonsente di riscrivere la (6.23) come

$$\boldsymbol{x} \in \mathfrak{C} \iff \boldsymbol{s}(\boldsymbol{x}) = \boldsymbol{0} \tag{6.28}$$

La dimensione della sindrome è $n - k$ e ci sono dunque q^{n-k} sindromi distinte; gli '0' della sindrome specificano le posizioni nelle quali le $n - k$ equazioni di

controllo vengono soddisfatte. Se in tutte le posizioni c'è 0, allora tutte le equazioni di controllo sono soddisfatte e la parola appartiene al codice. Poiché le parole che possono uscire dal canale sono q^n mentre le sindromi sono in tutto q^{n-k} non c'è corrispondenza biunivoca (p.es. tutte le parole di codice hanno sindrome nulla). Si noti però che, per effetto della linearità dello spazio vettoriale, si ricava

$$\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{H} \cdot \boldsymbol{y}^T = \boldsymbol{H} \cdot \left(\boldsymbol{x}^T + \boldsymbol{e}^T\right) = \boldsymbol{H} \cdot \boldsymbol{x}^T + \boldsymbol{H} \cdot \boldsymbol{e}^T = \boldsymbol{H} \cdot \boldsymbol{e}^T = \boldsymbol{s}(\boldsymbol{e}) \tag{6.29}$$

in quanto la sindrome di una parola di codice è nulla. Dunque vettore ricevuto e vettore d'errore hanno la stessa sindrome. Lo scopo nella decodifica con correzione dell'errore è allora quello di individuare $\boldsymbol{e}$ conoscendo $\boldsymbol{s}(\boldsymbol{e})$.

Prima di affrontare questo problema illustriamo la procedura di decodifica nella quale ci si accontenti di rivelare l'errore.

Procedura di decodifica con sindrome (rivelazione d'errore).

1. Si riceve $\boldsymbol{y}$ dal canale e si calcola la sindrome $\boldsymbol{s}(\boldsymbol{y})$;
2. Se $\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{0} \implies \widehat{\boldsymbol{x}} = \boldsymbol{y}$;
3. Se $\boldsymbol{s}(\boldsymbol{y}) \neq \boldsymbol{0} \implies$ rivelazione d'errore!

Se si giunge al passo 3 ci si può naturalmente fermare, e chiedere la ritrasmissione del dato. Cerchiamo ora una strategia per correggere l'errore. Se la sindrome del vettore ricevuto non è nulla si ha $\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{s}(\boldsymbol{x} + \boldsymbol{e}) = \boldsymbol{s}(\boldsymbol{e}) \neq \boldsymbol{0}$; ma $\boldsymbol{s}(\boldsymbol{x_i} + \boldsymbol{e}) = \boldsymbol{s}(\boldsymbol{y})$ per ogni parola di codice $\boldsymbol{x_i}$, e dunque le n-ple che hanno la stessa sindrome sono q^k. Costruiamo allora la seguente *tabella di Slepian*

$\boldsymbol{0}$	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{\ldots}$	$\boldsymbol{x_{q^k-1}}$	$\boldsymbol{s_0 = 0}$	
$\boldsymbol{e_1}$	$\boldsymbol{x_1 + e_1}$	$\boldsymbol{x_2 + e_1}$	$\boldsymbol{\ldots}$	$\boldsymbol{x_{q^k-1} + e_1}$	$\boldsymbol{s_1 = s(e_1)}$	$\Big\}\, q^{n-k}$
$\boldsymbol{e_2}$	$\boldsymbol{x_1 + e_2}$	$\boldsymbol{x_2 + e_2}$	$\boldsymbol{\ldots}$	$\boldsymbol{x_{q^k-1} + e_2}$	$\boldsymbol{s_2 = s(e_2)}$	
$\vdots$			$\vdots$		$\vdots$	
$\vdots$			$\vdots$		$\vdots$	

disponendo tutte le q^k parole di codice sulla prima riga; esse costituiscono un *sottogruppo*. Le righe rimanenti vengono riempite scegliendo successivamente un vettore $\boldsymbol{e_j}$ non precedentemente incontrato, da inserire nella prima colonna, e riportando sulla stessa riga le somme del tipo $\boldsymbol{e_j} + \mathfrak{C}$. Ciascuna riga costituisce allora un *laterale del sottogruppo* $\mathfrak{C}$ (vedi par. 6.2.1) e i vettori $\boldsymbol{e_j}$ sono i *generatori dei laterali* . Ricordiamo che due elementi $\boldsymbol{y}$ e $\boldsymbol{e}$ stanno sullo stesso laterale

(sono equivalenti) se e solo se la loro differenza è un elemento di $\mathfrak{C}$. Nel nostro caso possiamo specificare che

$$\boldsymbol{y}-\boldsymbol{e}\in\mathfrak{C}\Longleftrightarrow \boldsymbol{s}(\boldsymbol{y}-\boldsymbol{e})=\boldsymbol{0}\Longleftrightarrow \boldsymbol{s}(\boldsymbol{y})=\boldsymbol{s}(\boldsymbol{e}) \tag{6.30}$$

e dunque due elementi che stanno sullo stesso laterale hanno la stessa sindrome. Ma allora possiamo scegliere come generatori di laterali delle particolari n-ple d'errore, e stimare $\boldsymbol{x}$ direttamente dalla tabella cercando la parola di codice che sta nella stessa colonna di $\boldsymbol{y}$ ricevuto. Naturalmente la qualità della decodifica dipenderà dalla scelta dei generatori dei laterali, cioè delle n-ple d'errore, che dovranno essere quelle le più verosimili nel corso della trasmissione. Poiché le n-ple d'errore più probabili sono quelle con peso più basso, per rendere minima la probabilità d'errore sarà opportuno scegliere queste ultime come generatori di laterali.

Esempio 6.7. Sia $n=4$, $k=2$ e $\boldsymbol{G}=\begin{pmatrix}0&1&1&0\\1&0&1&1\end{pmatrix}$. Costruiamo la tabella di Slepian corrispondente

0000	0110	1011	1101	00
1111	1001	0100	0010	11
1000	1110	0011	0101	10
0111	0001	1100	1010	01

scegliendo a caso i generatori dei laterali. L'ultima colonna contiene il trasposto della sindrome relativa a ciascun laterale. La matrice di controllo $\boldsymbol{H}$ si ricava individuando due vettori ortogonali alle righe della $\boldsymbol{G}$, p.es $\boldsymbol{H}=\begin{pmatrix}1&1&1&0\\0&1&1&1\end{pmatrix}$. La struttura della matrice suggerisce subito che gli errori in 2^a e 3^a posizione sono indistinguibili, poiché le due colonne corrispondenti sono uguali. Ciò significa che 0100 e 0010 hanno la stessa sindrome e stanno dunque nello stesso laterale. Supponiamo di trasmettere $\boldsymbol{x}=0110$; la decodifica avviene in modo corretto solo se l'ennupla d'errore che si sovrappone a $\boldsymbol{x}$ è un generatore dei laterali, dunque se $\boldsymbol{e}=0000$ oppure $1111, 1000, 0111$. In tutti gli altri casi si verificherà errore. Se per esempio riceviamo $\boldsymbol{y}=0111$, la sindrome vale 01^T e la stima che facciamo è $\widehat{\boldsymbol{x}}=0000$, che è sbagliata in quanto l'ennupla d'errore $0001=\boldsymbol{y}-\boldsymbol{x}$ non è generatore di laterali. Come anticipato sarà opportuno usare come generatori delle ennuple leggere, poiché quelle pesanti hanno una bassa probabilità di realizzarsi, ed è quindi bassa la probabilità di correggere l'errore.

Una scelta migliore per i generatori è la seguente

0000	0110	1011	1101	00
0100	0010	1111	1001	11
1000	1110	0011	0101	10
0001	0111	1010	1100	01

In questo modo vengono corrette le configurazioni d'errore con un errore in 1^a, 2^a e 4^a posizione. Un errore in terza posizione non è invece correggibile, poiché 0010 non è generatore di laterale. Viceversa imponendo 0010 come generatore si escluderebbe 0100, che sta comunque sullo stesso laterale; ciò è legato alla struttura della matrice di controllo e al fatto che la 2^a e 3^a colonna sono uguali. ◯

Procedura di decodifica con sindrome (correzione d'errore).

1. Si riceve $\boldsymbol{y}$ dal canale e si calcola la sindrome $\boldsymbol{s}(\boldsymbol{y})$.
2. Se $\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{0} \quad \Longrightarrow \quad \widehat{\boldsymbol{x}} = \boldsymbol{y}$.
3. Se $\boldsymbol{s}(\boldsymbol{y}) \neq \boldsymbol{0}$ si legge $\boldsymbol{e}$, generatore del laterale, e si pone $\widehat{\boldsymbol{x}} = \boldsymbol{y} - \boldsymbol{e}$.

Oss. 6.3. Il fatto che la sindrome sia nulla, e che quindi l'n-pla appartenga al codice, non implica che non si siano verificati errori sul canale. Anzi, se ce ne sono il peso di $\boldsymbol{e}$ sarà probabilmente rilevante, tale cioè da trasformare una parola di codice in un'altra parola di codice.

La probabilità di corretta decodifica si ricava direttamente dal peso dei generatori dei laterali

$$P_C = \sum_{w=0}^{n} n_w \epsilon^w (1-\epsilon)^{n-w} \tag{6.31}$$

con n_w pari al numero di generatori di peso w .

A seguito di quanto detto finora si deduce che le prestazioni del codice dipendono dalla struttura dei generatori; se vogliamo correggere tutte le configurazioni di fino a t errori, dovranno essere presenti come generatore del laterale tutte le $\binom{n}{w}(q-1)^w$ configurazioni di peso w con $0 \leq w \leq t$, dove il termine $(q-1)^w$ deriva dal fatto che in corrispondenza di ciascuna posizione d'errore ci sono $q-1$ lettere dell'alfabeto q-ario (tutte meno la nulla) da inserire come errore possibile. Ecco come si presenta in questo caso la colonna dei generatori dei laterali

$00\ldots0$	(n-pla di peso 0)	$\ldots$	$\ldots$	$\Big\}\ q^{n-k}$
$\binom{n}{1}(q-1)$	n-ple di peso 1	$\ldots$	$\ldots$	
$\binom{n}{2}(q-1)^2$	n-ple di peso 2	$\ldots$	$\ldots$	
$\vdots$		$\ldots$	$\ldots$	
$\binom{n}{t}(q-1)^t$	n-ple di peso t	$\ldots$	$\ldots$	
altri generatori di peso $> t$		$\ldots$	$\ldots$	

Se il codice corregge tutte *e sole* le configurazioni fino a t errori (codice perfetto), non ci sono altri generatori di laterali oltre a quelli di peso t, e la colonna finisce in corrispondenza di questi ultimi; altrimenti ci sono anche alcuni generatori di peso $w > t$, e il codice corregge anche queste configurazioni.

Si noti che ciascuna colonna della tabella di Slepian, relativa a una certa parola di codice $\boldsymbol{x}$, contiene tutti i vettori che nella decodifica vengono associati a $\boldsymbol{x}$, costituendo la *regione di decodifica* $\mathcal{R}_{\boldsymbol{x}}$ di $\boldsymbol{x}$. Se il codice è t correttore tale regione conterrà quantomeno tutti i $\sum_{w=0}^{t} \binom{n}{w}(q-1)^w$ vettori che si trovano sui gusci a distanza compresa tra 0 e t, e che costituiscono la *sfera di Hamming* di raggio t centrata in $\boldsymbol{x}$, più eventualmente qualche altro vettore di peso maggiore di t ed esterno alla sfera. Viceversa, nel caso di codice perfetto non ci saranno altri vettori esterni, mancando i generatori di laterali di peso $> t$. La condizione di perfezione (6.16) per i codici lineari si esprime pertanto nel modo seguente

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i = q^{n-k} \tag{6.32}$$

Esempio 6.8. Si consideri il codice a ripetizione C(5,1) dell'esempio 6.2, la cui matrice generatrice è $\boldsymbol{G_1} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$. La tabella di Slepian corrispondente la si ricava a partire dalle due parole di codice 00000 e 11111, selezionando come generatori tutte le configurazioni di peso 1 e 2.

Si noti che $2\left[1 + \binom{5}{1} + \binom{5}{2}\right] = 2^5$ e il codice è perfetto. Esso corregge infatti

	00000	11111
$\binom{5}{1}$	10000 ⋮ 00001	01111 ⋮ 11110
$\binom{5}{2}$	11000 ⋮ 00011	00111 ⋮ 11100

tutte le configurazioni di errore di peso 1 e 2, ma *nessuna* di peso ≥ 3. ◯

6.3.3 Codici di Hamming

I codici di Hamming furono introdotti nel 1950 e costituiscono il primo esempio di impiego sistematico di tecniche algebriche per strutturare il dizionario del codice, rendendo disponibili tutte le sue parole mediante semplici calcoli matriciali. Le prestazioni dal punto di vista della capacità di correzione

non sono come vedremo esaltanti (sono codici 1-correttori); tuttavia la semplicità delle operazioni di codifica/decodifica, legata alla possibilità di una loro meccanizzazione, ne hanno decretato un significativo successo sul versante applicativo.

Consideriamo nuovamente la matrice di controllo $\boldsymbol{H_3}$ dell'esempio 6.6.

$$\boldsymbol{H_3} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Si è visto che non esistono coppie di colonne linearmente dipendenti (uguali nel caso binario in questione) e che il peso minimo delle parole di codice, e quindi la distanza minima, è pari a 3. Il codice può correggere dunque tutte le configurazioni di 1 errore. Tutto ciò si può generalizzare prendendo $m = 2, 3, 4, \ldots$ e costruendo una matrice $\boldsymbol{H}$ di controllo che non abbia coppie di colonne linermente dipendenti, usando tutte le possibili $n = 2^m - 1$ colonne m-ple distinte eccetto la nulla; in tal caso $m = n - k$ e $k = 2^m - 1 - m$. Così facendo si ottiene un codice di *Hamming* (binario) con parametri $C(2^m - 1, 2^m - 1 - m)$ che permette di correggere tutte le configurazioni di 1 errore, essendo $d_{min} = 3$. Nel caso generale con un alfabeto q-ario, per ciascuna delle $q^m - 1$ m-ple possibili ce ne sono $q - 2$ che sono linermente dipendenti. La matrice può allora essere costruita nel modo seguente. All'inizio si prenda una qualunque colonna non nulla $\boldsymbol{h_1}$ di $\boldsymbol{q^m}$; come seconda colonna si scelga $\boldsymbol{h_2}$ dall'insieme

$$\boldsymbol{q^m} - \{\alpha \boldsymbol{h_1} : \alpha \neq 0\} \tag{6.33}$$

Iterando la procedura ed eliminando a ogni passo dall'insieme $\boldsymbol{q^m}$ un numero di colonne pari a

$$|\{\alpha \boldsymbol{h_1} : \alpha \neq 0\}| = q - 1$$

si giunge alla fine a una matrice di controllo con $(q^m - 1)/(q - 1)$ colonne, per la quale non esistono coppie di colonne linearmente dipedenti, ma esistono terne di colonne che lo sono. I parametri di un codice di Hamming q-ario sono allora

$$n = \frac{q^m - 1}{q - 1} \quad k = \frac{q^m - 1}{q - 1} - m \quad d_{min} = 3 \tag{6.34}$$

Per quanto riguarda la decodifica, la presenza di tutte le possibili colonne (esclusa la nulla) nella matrice di controllo semplifica notevolmente la procedura, consentendo di evitare la costruzione della tabella di Slepian. Infatti, ricevuto $\boldsymbol{y}$ dalle (6.26) e (6.29) possiamo scrivere

$$\boldsymbol{H} \cdot \boldsymbol{y}^T = \boldsymbol{H} \cdot \boldsymbol{e}^T = \sum_{e_i \neq 0} e_i \boldsymbol{h_i} \tag{6.35}$$

e dunque la sindrome corrisponde alla combinazione lineare delle colonne di $\boldsymbol{H}$ in corrispondenza delle quali si è verificato l'errore (di ampiezza e_i). Se non ci sono errori la sindrome è nulla e si stima $\widehat{\boldsymbol{x}} = \boldsymbol{y}$. Se si è verificato un solo errore in posizione i la sindrome vale $e_i \boldsymbol{h_i}$, ed è dunque proporzionale (uguale nel caso binario) alla colonna di $\boldsymbol{H}$ posizionata in corrispondenza dell'errore. Per semplificare la procedura, nel caso binario si possono ordinare le colonne di $\boldsymbol{H}$ in modo che rappresentino la codifica binaria della posizione della colonna; così facendo la sindrome fornisce direttamente la posizione dell'errore espressa in base 2.

Esempio 6.9. Prendiamo $m = 3$ e costruiamo la matrice di controllo per il codice di Hamming binario C(7,4), ordinando le colonne come specificato.

$$\boldsymbol{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{6.36}$$

Se si verifica un errore in posizione 3 la sindrome è data da 011^T, che letto in binario indica la necessità di correggere un errore in terza posizione. ◯

Nel caso q-ario possiamo costruire la matrice organizzando le colonne per valori crescenti dei numeri rappresentati in base q; avendo inoltre l'accortezza di usare sempre un 1 come primo elemento di ciascuna colonna, in fase di decodifica si conoscerà immediatamente l'ampiezza di e_i, che verrà letta in corrispondenza del primo elemento non nullo della sindrome.

Esempio 6.10. Prendiamo nuovamente $m = 3$ e $q = 3$. Usando la procedura vista nella (6.33) e tenendo conto delle osservazioni fatte costruiamo la seguente matrice

$$\boldsymbol{H} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$$

del codice di Hamming ternario C(13,10). Supponiamo ora che ci sia un errore di ampiezza 2 in posizione 7. La sindrome che si ricava è $(201)^T$. Poiché c'è un 2 nella prima posizione questo corrisponde all'ampiezza, e la sindrome può essere scritta come $2 \cdot (102)^T$, che mette in evidenza la 7^a colonna della matrice. L'errore da correggere è dunque di ampiezza 2 in posizione 7. ◯

Quanto specificato finora porta alla seguente

Procedura di decodifica per un codice di Hamming.

1. Si riceve $\boldsymbol{y}$ dal canale e si calcola la sindrome $\boldsymbol{s}(\boldsymbol{y})$.
2. Se $\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{0} \implies \widehat{\boldsymbol{x}} = \boldsymbol{y}$.

3. Se $\boldsymbol{s(y)} = e_i\boldsymbol{h_i}$ si corregge la i-esima posizione con un'ampiezza e_i.

Vediamo quali sono le prestazioni offerte da questa procedura di decodifica:

$wt(\boldsymbol{e}) = 0$: Se $\boldsymbol{e} = \boldsymbol{0}$ si ha $\boldsymbol{y} = \boldsymbol{x}$ $\boldsymbol{s(y)} = \boldsymbol{0}$ $\widehat{\boldsymbol{x}} = \boldsymbol{x}$ $\Longrightarrow$ SÌ

$wt(\boldsymbol{e}) = 1$: Se $e_j \neq 0$ $\boldsymbol{s(y)} = e_j\boldsymbol{h_j}$, si modifica la j-esima posizione con un'ampiezza e_j, $\widehat{\boldsymbol{x}} = \boldsymbol{x}$ $\Longrightarrow$ SÌ

$wt(\boldsymbol{e}) = 2$: Se $e_i, e_j \neq 0$ si ha $\boldsymbol{s(y)} = e_i\boldsymbol{h_i} + e_j\boldsymbol{h_j} = e_r\boldsymbol{h_r}$. L'algoritmo indica di modificare la posizione r con un'ampiezza e_r e si introduce un terzo errore. $\Longrightarrow$ NO

$wt(\boldsymbol{e}) \geq 2$: In tal caso l'algoritmo non funziona più. $\Longrightarrow$ NO

Codice di Hamming esteso

Le prestazioni del codice di Hamming possono essere migliorate in modo significativo a spese di una lieve complicazione. L'idea è quella di aumentare la distanza minima da 3 a 4, individuando una procedura di decodifica che consenta contemporaneamente di correggere tutte le configurazioni di errore di peso 1 e di rivelare quelle di peso 2. L'estensione si avvale dell'introduzione di un ulteriore cifra di controllo che assicuri la *parità* della parola di codice.

Teorema 6.1. *Sia d_{min} dispari. Allora un codice binario con parametri (n, d_{min}) esiste se e solo se esiste un codice un codice $(n+1, d_{min}+1)$.*

Dim. Supponiamo che C sia un codice (n, d_{min}) binario. Estendiamo ora il codice aggiungendo un bit di parità a ciascuna sua parola

$$\boldsymbol{u} = (u_1u_2\ldots u_n) \quad \Longrightarrow \quad \widetilde{\boldsymbol{u}} = \begin{cases} u_1u_2\ldots u_n0 & \text{se } wt(\boldsymbol{u}) \text{ è pari} \\ u_1u_2\ldots u_n1 & \text{se } wt(\boldsymbol{u}) \text{ è dispari} \end{cases}$$

cioè in modo tale che sia $u_{n+1} = \sum_{i=1}^{n} u_i \mod 2$ (si aggiunge 0 se il peso è pari e 1 se è dispari). Il nuovo codice esteso $\widetilde{C}$ è formato solamente da parole pari, e poiché assegnate $\boldsymbol{u}, \boldsymbol{v} \in \widetilde{C}$ si ha (nel caso binario)

$$wt(\boldsymbol{u} \oplus \boldsymbol{v}) = wt(\boldsymbol{u}) + wt(\boldsymbol{u}) - 2wt(\boldsymbol{u} \cdot \boldsymbol{v}) \tag{6.37}$$

come si può dimostrare facilmente, se ne deduce che il peso della somma, e dunque la $d_{min}(\widetilde{C})$, è ancora pari. Chiaramente $d_{min} \leq d_{min}(\widetilde{C}) \leq d_{min}+1$ ed essendo d_{min} dispari deve essere $d_{min}(\widetilde{C}) = d_{min}+1$, e quindi $\widetilde{C}$ è un $(n+1, d_{min}+1)$-codice. Viceversa supponiamo che $\widetilde{C}$ sia un codice $(n+1, d_{min}+1)$-codice con d dispari. Prendiamo due parole $\widetilde{\boldsymbol{u}}$ e $\widetilde{\boldsymbol{v}}$ tali che sia

proprio $d_H(\widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{v}}) = d_{min}+1$. Scegliamo una posizione in cui $\widetilde{\boldsymbol{u}}$ e $\widetilde{\boldsymbol{v}}$ differiscono e cancelliamola da tutte le parole di codice. Il codice risultante è un (n, d_{min})-codice □

Sfruttando il teorema 6.1 si può passare da un codice $C(n,k,3)$ a uno $C(n+1,k,4)$.

La struttura generale della matrice di controllo del *codice di Hamming esteso* è la seguente

$$\widehat{\boldsymbol{H}} = \left(\begin{array}{ccccccc|c} 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 \\ \hline & & & & & & & 0 \\ & & & & & & & 0 \\ & & & \boldsymbol{H} & & & & \vdots \\ & & & & & & & 0 \\ & & & & & & & 0 \end{array}\right)$$

dove la prima riga corrisponde all'equazione di controllo $\sum_{j=1}^{n+1} y_j = 0$ della parità della parola di codice, con $\boldsymbol{H} = (\boldsymbol{h_1 h_2 \dots h_n})$ matrice binaria di Hamming. Vediamo l'algoritmo di decodifica

Procedura di decodifica per un codice di Hamming esteso.

1. Si riceve $\boldsymbol{y}$ dal canale e si calcola la sindrome $\boldsymbol{s}(\boldsymbol{y}) = \begin{pmatrix} \alpha \\ \boldsymbol{h_i} \end{pmatrix}$.

2. Se $\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{0} \implies \widehat{\boldsymbol{x}} = \boldsymbol{y}$.

3. Se $\boldsymbol{s}(\boldsymbol{y}) \neq \boldsymbol{0}$ e $\alpha = 1$ allora consideriamo la seconda parte della sindrome $\boldsymbol{h_i}$, la leggiamo in binario e modifichiamo la i-esima posizione (la posizione $n+1 = 2^m$ se $\boldsymbol{h_i}$ è la colonna nulla).

4. Se $\boldsymbol{s}(\boldsymbol{y}) \neq \boldsymbol{0}$ e $\alpha = 0$ dichiariamo errore.

Per quanto riguarda le prestazioni offerte da questa procedura di decodifica si ottiene

$wt(\boldsymbol{e}) = 0$: Se $\boldsymbol{e} = \mathbf{0}$ allora $\boldsymbol{y} = \boldsymbol{x}$, $\boldsymbol{s}(\boldsymbol{y}) = \mathbf{0}$, $\widehat{\boldsymbol{x}} = \boldsymbol{x}$ $\Longrightarrow$ SÌ

$wt(\boldsymbol{e}) = 1$: Se $e_j \neq 0$ si ha $\boldsymbol{s}(\boldsymbol{y}) = \begin{pmatrix} 1 \\ \boldsymbol{h_i} \end{pmatrix}$ e quindi $\alpha = 1$, si modifica la j-esima posizione correggendo l'errore $\Longrightarrow$ SÌ

$wt(\boldsymbol{e}) = 2$: Se $e_i, e_j \neq 0$ la sindrome vale $\boldsymbol{s}(\boldsymbol{y}) = \begin{pmatrix} 0 \\ \boldsymbol{h_r} \end{pmatrix}$, si ha $\alpha = 0$ e si riconosce la presenza dell'errore doppio $\Longrightarrow$ SÌ

$wt(\boldsymbol{e}) = 3$: si ottiene $\boldsymbol{s}(\boldsymbol{y}) \neq \mathbf{0}$ con $\alpha = 1$, si modifica una posizione sbagliata e si introduce un altro errore $\Longrightarrow$ NO

$wt(\boldsymbol{e}) = 4$: come il caso per $wt(\boldsymbol{e}) = 2$. $\Longrightarrow$ SÌ

La procedura vista consente di correggere gli errori di peso 1 e di rivelare tutte le configurazioni d'errore di peso pari. Naturalmente si potrebbe usare il codice per rivelare tutte le configurazioni fino a 3 errori, chiedendo la ritrasmissione dei dati quando la sindrome è nulla.

Le prestazioni asintotiche del codice di Hamming sono tuttavia sconfortanti dal punto di vista della capacità di correzione

$$\lambda_{_H} = \frac{d_{min}}{n} = \frac{3}{n} \longrightarrow 0 \tag{6.38}$$

poichè la distanza minima non dipende dall'ordine m del codice e rimane bloccata sul valore 3. Viceversa il tasso è asintoticamente unitario

$$R_{_H} = \frac{k}{n} = \frac{2^m - 1 - m}{2^m - 1} \longrightarrow 1 \tag{6.39}$$

6.3.4 Codici BCH

I codici BCH furono ideati indipendentemente da A. Hocquenghem (1959) e da R.C. Bose e D.K. Ray-Chaudhuri (1960). Essi costituiscono la naturale generalizzazione del codice di Hamming e consentono di correggere una qualunque configurazione fino a t errori.

In questo capitolo studieremo solamente il caso binario con $t = 2$, che può essere introdotto in modo euristico senza coinvolgere in modo massiccio la struttura dei campi di Galois $GF(p^m)$.

Partiamo nuovamente dalla matrice $m \times n$ di controllo $\boldsymbol{H} = (\boldsymbol{h_1 h_2 \ldots h_n})$ per un codice di Hamming 1-correttore, con parametri $m = n - k, n = 2^m - 1$ e $k = 2^m - 1 - m$. Se con m equazioni di controllo si corregge 1 errore, la speranza è che con $2m$ equazioni sia possibile correggerne 2. A tal scopo raddoppiamo la matrice di controllo

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H_1} \\ \text{---} \\ \boldsymbol{H_2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{h_1} & \boldsymbol{h_2} & \ldots & \ldots & \boldsymbol{h_n} \\ \text{----------} & & & & \\ \boldsymbol{k_1} & \boldsymbol{k_2} & \ldots & \ldots & \boldsymbol{k_n} \end{pmatrix} \tag{6.40}$$

dove le $\boldsymbol{h_j}$ sono le m-ple binarie della matrice $\boldsymbol{H_1}$ di Hamming e le $\boldsymbol{k_j}$ sono m-ple binarie da costruire in funzione delle $\boldsymbol{h_j}$. Porremo dunque

$$\boldsymbol{k_j} = f(\boldsymbol{h_j})$$

con $f(.)$ funzione da determinare. Si noti intanto che, ricevuto $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$, il calcolo della sindrome porta a un vettore di dimensione $2m$, e quindi una sindrome superiore (la vecchia relativa al codice di Hamming) e una sindrome inferiore (la nuova che dipenderà dalla scelta delle $\boldsymbol{k_j}$)

$$\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{H}\boldsymbol{y}^T = \begin{pmatrix} \boldsymbol{s_1}(\boldsymbol{y}) \\ \text{- - - - -} \\ \boldsymbol{s_2}(\boldsymbol{y}) \end{pmatrix}$$

Vediamo cosa dovrebbe succedere per diversi pesi dell'ennupla d'errore:

1. Se $wt(\boldsymbol{e}) = \boldsymbol{0}$ si ha $\boldsymbol{s_1}(\boldsymbol{y}) = \boldsymbol{0}$, e vogliamo che sia anche $\boldsymbol{s_2}(\boldsymbol{y}) = \boldsymbol{0}$.
2. Se $wt(\boldsymbol{e}) = \boldsymbol{1}$, con $e_i = 1$, si ha $\begin{cases} \boldsymbol{s_1}(\boldsymbol{y}) = \boldsymbol{h_i} \\ \\ \boldsymbol{s_2}(\boldsymbol{y}) = f(\boldsymbol{h_i}) \end{cases}$

   In questo caso l'informazione $\boldsymbol{s_2}(\boldsymbol{y})$ è inutile, ma bisogna accorgersi che si è di fronte a 1 errore.
3. se $wt(\boldsymbol{e}) = \boldsymbol{2}$, con $e_i, e_j = 1$, si ha $\begin{cases} \boldsymbol{s_1}(\boldsymbol{y}) = \boldsymbol{h_i} + \boldsymbol{h_j} \\ \\ \boldsymbol{s_2}(\boldsymbol{y}) = f(\boldsymbol{h_i}) + f(\boldsymbol{h_j}) \end{cases}$

Il problema è allora quello di ricavare i e j, cioè le posizioni degli errori, a partire dal sistema impostato; ciò viene fatto scegliendo $f(.)$ in modo opportuno. Scelte tipo $f(\boldsymbol{h_i}) = \boldsymbol{h_i}$ oppure $f(\boldsymbol{h_i}) = \boldsymbol{h_i} + \boldsymbol{\alpha}$ con $\boldsymbol{\alpha}$ costante sono inutili: infatti la prima non porta informazione suppletiva, ma neanche la seconda, poiché essendo $\boldsymbol{\alpha} + \boldsymbol{\alpha} = \boldsymbol{0}$ su $GF(2)$ si avrebbe $\boldsymbol{s_1}(\boldsymbol{y}) = \boldsymbol{s_2}(\boldsymbol{y})$. Anche $f(\boldsymbol{h_i}) = \boldsymbol{h_i}^2$ non porta a nulla, in quanto si otterrebbe

$$\boldsymbol{s_1}(\boldsymbol{y})^2 = (\boldsymbol{h_i} + \boldsymbol{h_j})^2 = \boldsymbol{h_i}^2 + \boldsymbol{h_j}^2 + 2\boldsymbol{h_i}\boldsymbol{h_j} = \boldsymbol{s_2}(\boldsymbol{y})$$

poiché su $GF(2)$ accade che $2 \equiv 0$.

Con $f(\boldsymbol{h_i}) = \boldsymbol{h_i}^3$ si trova invece una possibile soluzione; poniamo per semplicità $\boldsymbol{h_i} = \boldsymbol{u}$, $\boldsymbol{h_j} = \boldsymbol{v}$, $\boldsymbol{s_1}(\boldsymbol{y}) = \boldsymbol{s_1}$ e $\boldsymbol{s_2}(\boldsymbol{y}) = \boldsymbol{s_3}$ (il 3 come pedice per ricordare che è riferito al caso di elevamento al cubo). Per le due sindromi vale la relazione

$$\begin{cases} \boldsymbol{s_1} = \boldsymbol{u} + \boldsymbol{v} \qquad (\boldsymbol{s_1} \neq \boldsymbol{0}) \\ \\ \boldsymbol{s_3} = \boldsymbol{u}^3 + \boldsymbol{v}^3 \end{cases}$$

Si noti però che

$$\boldsymbol{s_1}^3 = (\boldsymbol{u}+\boldsymbol{v})^3 = \boldsymbol{u}^3 + \boldsymbol{u}^2\boldsymbol{v} + \boldsymbol{u}\boldsymbol{v}^2 + \boldsymbol{v}^3 = \boldsymbol{u}^3 + \boldsymbol{v}^3 + \boldsymbol{u}\boldsymbol{v}(\boldsymbol{u}+\boldsymbol{v})$$

cioè

$$\boldsymbol{s_1}^3 = \boldsymbol{s_3} + \boldsymbol{u}\boldsymbol{v}\boldsymbol{s_1} \quad \text{e quindi } \boldsymbol{u}\boldsymbol{v} = \left(\boldsymbol{s_1}^3 + \boldsymbol{s_3}\right)\boldsymbol{s_1}^{-1} = \boldsymbol{s_1}^2 + \boldsymbol{s_3}\boldsymbol{s_1}^{-1}$$

Si ha pertanto

$$\begin{cases} \boldsymbol{u}+\boldsymbol{v} = \boldsymbol{s_1} \\ \boldsymbol{u}\boldsymbol{v} = \boldsymbol{s_1}^2 + \boldsymbol{s_3}\boldsymbol{s_1}^{-1} \end{cases}$$

che è un sistema che sappiamo risolvere poichè se Σ e Π sono rispettivamente la somma e il prodotto di due quantità incognite, allora tali quantità sono soluzione dell'equazione $\boldsymbol{z}^2 - \Sigma\boldsymbol{z} + \Pi = 0$, cioè

$$\boldsymbol{z}^2 - (\boldsymbol{u}+\boldsymbol{v})\boldsymbol{z} + \boldsymbol{u}\boldsymbol{v} = 0 \tag{6.41}$$

Risolvendo l'equazione

$$\boldsymbol{z}^2 + \boldsymbol{s_1}\boldsymbol{z} + \left(\boldsymbol{s_1}^2 + \boldsymbol{s_3}\boldsymbol{s_1}^{-1}\right) = 0 \qquad (\boldsymbol{s_1} \neq \boldsymbol{0}) \tag{6.42}$$

cioè uguagliando a zero il *polinomio locatore degli errori*, si individuano le posizioni degli errori. I risultati ottenuti sono coerenti con i vincoli imposti; infatti

1. Se $wt(\boldsymbol{e}) = 0$ si ha $\boldsymbol{s_1} = \boldsymbol{s_3} = \boldsymbol{0}$, la sindrome è nulla e ciò corrisponde al fatto che non sono stati commessi errori.

2. Se $wt(\boldsymbol{e}) = 1$ si ha $\boldsymbol{s_1} = \boldsymbol{u}$ e $\boldsymbol{s_3} = \boldsymbol{u}^3$; sostituendo si ottiene $\boldsymbol{z}^2 + \boldsymbol{u}\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{z}+\boldsymbol{u}) = 0$, cioè $\boldsymbol{z_1} = \boldsymbol{0}$ e $\boldsymbol{z_2} = \boldsymbol{u}$.

3. Se $wt(\boldsymbol{e}) = 2$ si ha il polinomio completo e bisogna risolvere l'equazione.

La procedura di decodifica con sindrome per un codice BCH diventa allora la seguente:

Decodifica con sindrome per codici BCH 2-correttori.

1. Se $\boldsymbol{s_1} = \boldsymbol{s_3} = \boldsymbol{0}$ la sindrome è nulla e si decide che non sono stati commessi errori.

2. Se $\boldsymbol{s_1} = \boldsymbol{u} \neq \boldsymbol{0}$ $\boldsymbol{s_3} = \boldsymbol{u}^3$ si corregge la posizione $\boldsymbol{u}$.

3. Se $\boldsymbol{s_1} \neq \boldsymbol{0}$ e $\boldsymbol{s_3} \neq \boldsymbol{s_1}^3$ le soluzioni della (6.41) forniscono la posizione degli errori.

Il polinomio locatore degli errori (6.41) ha come variabili degli elementi che sono vettori binari m-dimensionali. Ora bisogna dare un significato alle operazioni di somma e prodotto (e quindi elevamento a potenza) tra questi vettori. L'interpretazione immediata è quella di associare ciascun vettore alla rappresentazione degli elementi di un campo di Galois $GF(2^m)$, che analizzaremo meglio nel paragrafo 6.4.1. Per il momento usiamo un'altra interpretazione più diretta, legata comunque alla precedente, e che fa riferimento ai *polinomi a coefficienti su* $GF(2)$ di grado $< m$. Assegnata infatti una certa m-pla $a_0, a_1, \ldots, a_{m-1}$ di elementi su $GF(q)$ (anche se per il momento considereremo q=2) possiamo stabilire la seguente associazione

$$\boldsymbol{a} = (a_0, a_1, \ldots, a_{m-1}) \quad \Longleftrightarrow \quad a(x) = a_0 + a_1 x + \ldots + a_{m-1} x^{m-1} \tag{6.43}$$

Per la somma tra vettori valgono allora le consuete regole che riguardano la somma tra polinomi, che viene eseguita $\mod 2$

$$\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{c} \quad \Longleftrightarrow \quad a(x) + b(x) = c(x)$$

Il prodotto va invece fatto mediante *riduzione modulo* $m(x)$, dove $m(x)$ è un *polinomio irriducibile di grado* m.

Per costruire la seconda parte della matrice $\boldsymbol{H}$ dobbiamo scegliere un polinomio $m(x)$ col quale fare le riduzioni.

Esempio 6.11. Sia $m = 3$ e si voglia costruire la matrice $\boldsymbol{H_2}$ della 6.40. Per la $\boldsymbol{H_1}$ si ha

$$\boldsymbol{H_1} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Scegliamo il polinomio di terzo grado $m(x) = x^3 + x + 1$, irriducibile su $GF(2)$ in quanto $m(0) = m(1) = 1 \neq 0$. Associando i vettori ai rispettivi polinomi secondo quanto stabilito dalla (6.43) si ottiene

$$\boldsymbol{H_1} = \begin{pmatrix} & & & x^2 & x^2 & x^2 & x^2 \\ & & & & & + & + \\ & x & x & & & x & x \\ & & + & & + & & + \\ 1 & & 1 & & 1 & & 1 \end{pmatrix}$$

Calcoliamo ora il cubo dei vettori colonna, tenendo conto che si impone la riduzione $\mod m(x)$, cioè che $x^3 + x + 1 = 0$.

1. $1^3 = 1$

2. $x^3 = x + 1$
3. $(x+1)^3 = x^3 + x^2 + x + 1 = x^2$
4. $(x^2)^3 = (x^3)^2 = (x+1)^2 = x^2 + 1$
5. $(x^2+1)^3 = x^6 + x^4 + x^2 + 1 = \ldots = x^2 + x$
6. $(x^2+x)^3 = x^6 + x^5 + x^4 + x^3 = \ldots = x^2 + x + 1$
7. $(x^2+x+1)^3 = \ldots = x$

La matrice $\boldsymbol{H}$ si completa tenendo conto dei cubi appena calcolati

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H_1} \\ \text{---} \\ \boldsymbol{H_2} \end{pmatrix} = \left(\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right)$$

Supponiamo ora di aver ricevuto un vettore $\boldsymbol{y}$ affetto da due errori in posizione $i = 3$ e $j = 6$. La sindrome è data dalla somma della 3^a e 6^a colonna

$$\boldsymbol{s}(\boldsymbol{y}) = \boldsymbol{s}(\boldsymbol{e}) = \begin{pmatrix} \boldsymbol{s_1} \\ \text{---} \\ \boldsymbol{s_3} \end{pmatrix} = \left(\begin{array}{c} 1 \\ 0 \\ 1 \\ \hline 0 \\ 1 \\ 1 \end{array}\right)$$

Ponendo $z = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$, che corrisponde al polinomio incognito $\alpha x^2 + \beta x + \gamma$, si deve risolvere l'equazione vettoriale

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}^2 + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^2 + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^{-1} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

con α, β, γ incognite. Essa corrisponde all'equazione polinomiale

$$(\alpha x^2 + \beta x + \gamma)^2 + (x^2+1)(\alpha x^2 + \beta x + \gamma) + (x^2+1)^2 + (x+1)(x^2+1)^{-1}$$

I prodotti vanno fatti nel modo descritto precedentemente, mediante riduzione $\bmod m(x)$, mentre per trovare il reciproco di un polinomio assegnato $p(x)$, operazione sempre possibile se $m(x)$ è irriducibile, bisogna individuare un polinomio

ax^2+bx+c tale che $(ax^2+bx+c)p(x)=1$. Nel nostro caso dobbiamo trovare $(x^2+1)^{-1}$, e dunque deve essere $(ax^2+bx+c)(x^2+1)=1$. Facendo il prodotto e riducendo $\mod x^3+x+1$ si arriva all'equazione $cx^2+ax+b+c=1$, che porta a $a=0, b=1, c=0$, cioè il polinomio x. Infatti $x(x^2+1)=x^3+x=x+1+x=1$. La soluzione dell'equazione si ottiene risolvendo il seguente sistema

$$\begin{cases} \alpha+\beta+\gamma=0 \\ \beta=1 \end{cases} \qquad \text{ottenendo le 2 soluzioni} \qquad \begin{cases} \alpha=0 \\ \beta=1 \\ \gamma=1 \end{cases} \qquad \begin{cases} \alpha=1 \\ \beta=1 \\ \gamma=0 \end{cases}$$

Il codice corregge 2 errori e dunque $d_{min} \geq 5$. È un codice C(7,1) e oltre alla parola nulla 0000000 ne ha solo un'altra. L'unica che soddisfa tutte le 6 equazioni di controllo è 1111111. Ha dunque le stesse parole di un codice a ripetizione, che è quindi un caso particolare di codice BCH. ◯

La procedura di decodifica, e in generale lo studio dei codici BCH t-correttori ($t > 2$), ricevono un impulso rilevante dalla comprensione della struttura dei campi finiti di Galois $GF(q^m)$, che come anticipato è legata all'interpretazione dei vettori m-ple come elementi del campo. Introduciamo tale struttura con un esempio relativo al caso $GF(2^3)$ appena analizzato.

Ritorniamo all'interpretazione (6.43) delle colonne m-ple della matrice $\boldsymbol{H_1}$ come polinomi di grado $< m$, con il prodotto eseguito mediante riduzione $\mod m(x)$ irriducibile su $GF(2)$, p.es. $m(x)=x^3+x+1$. Ricordiamo che per ogni m-pla si possono fare le operazioni di somma, prodotto e si può trovare l'inverso di una certa m-pla preassegnata. Ciò significa che i polinomi di grado $< m$, ridotti $\mod m(x)$ irriducibile, costituiscono un *campofinito*.

Consideriamo la successione $x^0, x^1, x^2, x^3, \ldots$ ed eseguiamo le riduzioni $\mod x^3+x+1$. Si ottiene la successione $1, x, x^2, x^3=x+1, x^4=x^3x=(x+1)x=x^2+x, \ldots$ evidenziata nella tabella sotto

$$\begin{aligned} &1 \\ &x \\ &x^2 \\ &x^3=x+1 \\ &x^4=x^3x=(x+1)x=x^2+x \\ &x^5=x^3x^2=(x+1)x^2=x^3+x^2=x^2+x+1 \\ &x^6=(x^3)^2=(x+1)^2=x^2+1 \\ &x^7=x^6x=(x^2+1)x=x^3+x=1 \\ &\vdots \end{aligned}$$

Si noti che con $x^7=1$ ricomincia un nuovo ciclo, nel quale i polinomi ridotti si ripresentano con lo stesso ordine. Se ora α è una *radice* di $m(x)=x^3+x+1$ (nell'estensione $GF(8)$ di $GF(2)$) si ha $\alpha^3+\alpha+1=0$, e dunque $\alpha^3=\alpha+1$.

Elementi di GF(8)		
0	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	$\alpha+1$	011
α^4	$\alpha^2+\alpha$	110
α^5	$\alpha^2+\alpha+1$	111
α^6	α^2+1	101

Tabella 6.1 Struttura del campo $GF(2^3)$.

Calcolando i polinomi di prima in α si ottiene la *rappresentazione degli elementi di GF(8)* descritta dalla tabella 6.1. Nella prima colonna gli elementi sono espressi come potenza di α, nella seconda come polinomi in α di grado < 3 e nella terza mediante i coefficienti dei corrispondenti polinomi. E' facile verificare che la struttura è quella di un campo e che gli inversi si possono calcolare direttamente esprimendo ciascun elemento come potenza di α. Per esempio, volendo calcolare il reciproco $(\alpha^2+1)^{-1}$ dell'esempio 6.11 si può evitare l'uso dei polinomi, e notare semplicemente dalla tabella che $\alpha^2+1=\alpha^6$; bisogna allora risolvere l'equazione $\alpha^6\alpha^i=1=\alpha^7$, che porta immediatamente a $i=1$ e $(\alpha^6)^{-1}=\alpha^{-6}=\alpha$. Poiché attraverso le potenze di α si ottengono tutti gli elementi del campo, α si definisce *radice primitiva del campo*. Con questa interpretazione possiamo ordinare le colonne di $\boldsymbol{H_1}$ *secondo le potenze di* α invece che secondo gli interi letti in base 2, inducendo un diverso ordinamento anche per le colonne di $\boldsymbol{H_2}$, che si ricava immediatamente elevando al cubo le colonne di $\boldsymbol{H_1}$ intese come potenze di α. La matrice $\boldsymbol{H}$ diventa allora

$$\boldsymbol{H}=\left(\begin{array}{ccccccc} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \hline 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \end{array}\right) \tag{6.44}$$

che corrisponde a

$$\boldsymbol{H}=\left(\begin{array}{ccccccc} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}\right) \tag{6.45}$$

Anche la decodifica è immediata. Riprendendo il caso dell'esempio 6.11 con gli errori in posizione $i = 3, j = 6$, si ottiene $\boldsymbol{s_1} = \alpha^3$ e $\boldsymbol{s_3} = \alpha^5$. Sostituendo nel polinomio locatore degli errori ricaviamo l'equazione

$$z^2 + \alpha^3 z + ((\alpha^3)^2 + \alpha^5 \alpha^{-3}) = z^2 + \alpha^3 z + 1 = (z + \alpha^2)(z + \alpha^5)$$

con le radici che corrispondono appunto alla 3^a e 6^a colonna.

Le prestazioni del codice BCH 2-correttore sono tuttavia deludenti, poiché asintoticamente equivalenti a quelle relative al codice di Hamming

$$\lambda_{_H} = \frac{5}{n} \longrightarrow 0 \qquad R_{_H} = \frac{2^m - 1 - 2m}{2^m - 1} \longrightarrow 1 \tag{6.46}$$

6.3.5 Codici non lineari di Hadamard

Faremo ora una breve digressione su una classe di codici *non lineari* molto importante, tanto dal punto di vista applicativo che da quello teorico. Il vincolo della linearità è stato introdotto all'inizio per dotare il codice di una struttura, con il fine di consentendo semplici operazioni di codifica/decodifica anche quando n diventa molto grande. Tuttavia, quando si desidera ottenere il massimo tasso a parità di capacità di correzione è necessario rinunciare alla linearità, per consentire di allocare le parole di codice nel modo più conveniente. Come esempio si può citare l'osservazione fatta in [59], che se vogliamo un codice con $n = 11$ che corregga 2 errori abbiamo a disposizione al massimo 16 parole di codice nel caso lineare, ma ben 24 nel caso non lineare. Per indicare un codice nel caso non lineare useremo la notazione (n, M, d_{min}), in quanto M non è più vincolato a essere la k-esima potenza di q.

Per la descrizione del codice di Hadamard useremo nuovamente delle matrici (le matrici di Hadamard, per l'appunto), anche se col semplice significato di *tabelle rappresentative delle parole di codice*. Introduciamo alcune definizioni.

Def. 6.6. *Assegnata una matrice $\boldsymbol{\Pi}$, quadrata, $n \times n$, essa si dice di Hadamard se è composta solamente dagli elementi $+1$ e -1 e se*

$$\boldsymbol{\Pi} \cdot \boldsymbol{\Pi}^T = n\boldsymbol{I_n} \tag{6.47}$$

con $\boldsymbol{I_n}$ matrice identica di ordine n.

È immediato verificare che

Proposizione 6.2. *Assegnata una qualunque matrice $\boldsymbol{\Pi}$ di Hadamard*

1. *$\boldsymbol{\Pi}$ è non singolare;*
2. *anche $\boldsymbol{\Pi}^T$ è di Hadamard;*

3. *è sempre possibile trasformare $\boldsymbol{\Pi}$ nella sua forma normale, in cui la prima riga e la prima colonna contengono solo +1.*

Dim. Per la 1) basta osservare che sulla base della (6.47) risulta $(\det \boldsymbol{\Pi})^2 = n^n$, cioè $\det \boldsymbol{\Pi} \neq 0$. Per la 2) si premoltiplichi la (6.47) per $\boldsymbol{\Pi}^{-1}$ ambo i membri, ottenendo $\boldsymbol{\Pi}^{-1} = 1/n\boldsymbol{\Pi}^T$; dunque $\boldsymbol{\Pi}^T(\boldsymbol{\Pi}^T)^T = n\boldsymbol{\Pi}^{-1}\boldsymbol{\Pi} = n\boldsymbol{I}$. Per il punto 3) si noti che moltiplicando una qualunque riga o colonna per -1 si ottiene sempre una matrice di Hadamard. Infatti $\boldsymbol{\Pi}\boldsymbol{\Pi}^T$ è in effetti un prodotto riga per riga. Dette allora $\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_n}$ le righe si ha

$$\boldsymbol{r_i}\boldsymbol{r_j}^T = \begin{cases} 0 & \text{se} \quad i \neq j \\ n & \text{se} \quad i = j \end{cases}$$

Supponiamo ora che la riga i venga moltiplicata per -1; si ha

$$-\boldsymbol{r_i}\boldsymbol{r_j}^T = \begin{cases} 0 & \text{se} \quad i \neq j \\ n & \text{se} \quad i = j \end{cases} \qquad (-\boldsymbol{r_i}(-\boldsymbol{r_i}^T))$$

□

Vediamo ora alcuni esempi per i primi valori di n. Per $n = 1$ si trova la $\boldsymbol{H_1} = (1)$ (che è banalmente nella forma normale) e la $\boldsymbol{H_1}' = (-1)$; entrambe soddisfano la relazione (6.47). Per costruire $\boldsymbol{H_2}$ poniamo la stessa nella sua forma normale

$$\boldsymbol{H_2} = \begin{pmatrix} 1 & 1 \\ 1 & x \end{pmatrix} \text{ con } x \text{ tale che } \begin{pmatrix} 1 & 1 \\ 1 & x \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & x \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

cioè $x = -1$. Le matrici di Hadamard di ordine 2 sono allora, oltre alla normale, tutte quelle che si ottengono moltiplicando per -1 una qualunque riga o colonna

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 \\ -1 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 \\ 1 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & 1 \\ -1 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

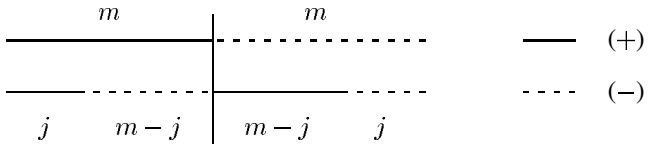
Per $\boldsymbol{H_3}$ si dovrebbe trovare una matrice

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & a & b \\ 1 & c & d \end{pmatrix}$$

tale che $\boldsymbol{\Pi}\boldsymbol{\Pi}^T = 3\boldsymbol{I}$, il che imporrebbe p.es. $1 + a + b = 0$, che non può essere soddisfatta con $a = \pm 1, b = \pm 1$. *Non esistono dunque matrici di Hadamard di ordine* 3. A tal riguardo si ha quest'importante teorema sulla struttura delle matrici di Hadamard

Teorema 6.2. *Tutte le matrici di Hadamard hanno ordine 1,2 o un multiplo di 4.*

Dim. Si costruisca una matrice di Hadamard di ordine n. Prendiamo la matrice nella sua forma normale e consideriamo la prima riga, che è fatta da soli 1; una seconda riga qualunque dovrà avere metà $+1$ e metà -1, poiché il prodotto scalare deve dare 0. Mediante alcune permutazioni di colonne, sempre possibili, possiamo disporre tutti i -1 nella parte finale. Poniamo allora $n = 2m$. Ripetiamo il ragionamento per la II riga rispetto alla terza; se nella prima metà di quest'ultima ci sono j elementi $+1$, per coerenza ci saranno $m - j$ elementi -1. Nella seconda metà sarà il viceversa, con $m - j$ di valore $+1$ e j di valore -1, poiché anche il prodotto tra la prima e la terza riga deve dare 0. Per l'ortogonalità tra II e III deve essere, con riferimento alla figura sotto



$$j - (m-j) - (m-j) + j = 0 \quad \text{cioè} \quad 4j = 2m = n$$

□

Oss. 6.4. Non è ancora noto se la condizione del teorema 6.2 sia anche sufficiente, e quindi se esistano matrici di Hadamard per ogni multiplo di 4.

Una matrice di Hadamard di ordine n può essere usata per costruirne una di ordine $2n$ secondo la seguente *costruzione di Sylvester*

$$\boldsymbol{H_{2n}} = \begin{pmatrix} \boldsymbol{\Pi} & \boldsymbol{\Pi} \\ \boldsymbol{\Pi} & \boldsymbol{-\Pi} \end{pmatrix}$$

Si verifica facilmente che anche $\boldsymbol{H_{2n}}$ è di Hadamard

$$\begin{aligned} \boldsymbol{H_{2n}}\boldsymbol{H_{2n}}^T &= \begin{pmatrix} \boldsymbol{\Pi} & \boldsymbol{\Pi} \\ \boldsymbol{\Pi} & \boldsymbol{-\Pi} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Pi}^T & \boldsymbol{\Pi}^T \\ \boldsymbol{\Pi}^T & \boldsymbol{-\Pi}^T \end{pmatrix} = \\ &= \begin{pmatrix} 2\boldsymbol{\Pi}\boldsymbol{\Pi}^T & \boldsymbol{\Pi}\boldsymbol{\Pi}^T - \boldsymbol{\Pi}\boldsymbol{\Pi}^T \\ \boldsymbol{\Pi}\boldsymbol{\Pi}^T - \boldsymbol{\Pi}\boldsymbol{\Pi}^T & 2\boldsymbol{\Pi}\boldsymbol{\Pi}^T \end{pmatrix} = \begin{pmatrix} 2n\boldsymbol{I_n} & 0 \\ 0 & 2n\boldsymbol{I_n} \end{pmatrix} = \\ &= 2n\boldsymbol{I_{2n}} \end{aligned}$$

Dunque, a partire da $\boldsymbol{H_1}$ si costruiscono immediatamente $\boldsymbol{H_2}$, $\boldsymbol{H_4}$, $\boldsymbol{H_8}$, $\boldsymbol{H_{16}}$, ... (scriviamo "-" al posto di "-1" per non appesantire la lettura)

$$\boldsymbol{H_1} = (1) \qquad \boldsymbol{H_2} = \begin{pmatrix} 1 & 1 \\ 1 & - \end{pmatrix}$$

$$\boldsymbol{H_4} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & - & 1 & - \\ 1 & 1 & - & - \\ 1 & - & - & 1 \end{pmatrix} \quad \boldsymbol{H_8} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & - & 1 & - & 1 & - & 1 & - \\ 1 & 1 & - & - & 1 & 1 & - & - \\ 1 & - & - & 1 & 1 & - & - & 1 \\ 1 & 1 & 1 & 1 & - & - & - & - \\ 1 & - & 1 & - & - & 1 & - & 1 \\ 1 & 1 & - & - & - & - & 1 & 1 \\ 1 & - & - & 1 & - & 1 & 1 & - \end{pmatrix}$$

Oltre alla costruzione secondo Sylvester ce ne sono altre per le quali rimandiamo a [59], tra cui quella secondo Paley che consente di ricavare matrici di ordine $n = p+1 = 4j$ con p primo; tale matrice la ritroveremo come nucleo costitutivo per la matrice generatrice del *codice binario di Golay* C(23, 2^{12}, 7) (si veda il paragrafo 6.3.6); col *metodo di Paley* si ottiene infatti la

$$\boldsymbol{H_{12}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & - & 1 & - & 1 & 1 & 1 & - & - & - & 1 & - \\ 1 & - & - & 1 & - & 1 & 1 & 1 & - & - & - & 1 \\ 1 & 1 & - & - & 1 & - & 1 & 1 & 1 & - & - & - \\ 1 & - & 1 & - & - & 1 & - & 1 & 1 & 1 & - & - \\ 1 & - & - & 1 & - & - & 1 & - & 1 & 1 & 1 & - \\ 1 & - & - & - & 1 & - & - & 1 & - & 1 & 1 & 1 \\ 1 & 1 & - & - & - & 1 & - & - & 1 & - & 1 & 1 \\ 1 & 1 & 1 & - & - & - & 1 & - & - & 1 & - & 1 \\ 1 & 1 & 1 & 1 & - & - & - & 1 & - & - & 1 & - \\ 1 & - & 1 & 1 & 1 & - & - & - & 1 & - & - & 1 \\ 1 & 1 & - & 1 & 1 & 1 & - & - & - & 1 & - & - \end{pmatrix}$$

che è equivalente a quella citata, cioè ottenibile mediante permutazioni di righe e/o colonne e moltiplicazione per -1.

Oss. 6.5. Per ciascuno dei valori n = 1, 2, 4, 8 e 12 esiste un'unica classe di equivalenza, e quindi tutte le matrici dello stesso ordine sono tra loro equivalenti. Le cose cambiano invece per $n = 16$, dove si possono individuare 5 classi di matrici che sono *strutturalmente* diverse le une dalle altre, e quindi non ottenibili mediante permutazioni di righe/colonne e moltiplicazioni per -1. Per $n = 20$ ci sono 3 classi e in generale il problema di conoscere il numero di classi per ciascun valore di n non è risolto.

Come anticipato, la struttura delle matrici di Hadamard viene sfruttata semplicemente come descrizione esplicita del dizionario di un certo codice (solitamente non lineare), descritto dal seguente

Teorema 6.3. *Assegnata una qualunque matrice* $\boldsymbol{H_n}$ *di Hadamard esiste un codice binario* $(n, M, d_{min}) = (n, 2n, n/2)$.

Dim. Si costruisca l'insieme dei $2n$ vettori $\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_n}, -\boldsymbol{r_1}, -\boldsymbol{r_2}, \ldots, -\boldsymbol{r_n}$ con $\boldsymbol{r_i}$ righe di $\boldsymbol{H_n}$. In ogni riga si cambi ora 1 con 0 e -1 con 1, ottenendo un insieme di $2n$ vettori binari. I vettori del tipo $\boldsymbol{r_i}$ e $-\boldsymbol{r_i}$ sono ovviamente a distanza n, mentre quelli del tipo $\pm\boldsymbol{r_i}$ sono ortogonali ai $\pm\boldsymbol{r_j}$ se $i \neq j$, e di conseguenza differiscono in $n/2$ posizioni e sono uguali nelle rimanenti $n/2$. Di conseguenza $d_{min} = n/2$. □

Oltre alla famiglia $\boldsymbol{\Pi}(n, 2n, n/2)$ dei codici descritti dal teorema se ne possono ricavare facilmente altre due con d_{min} dispari nel modo seguente:

$\boldsymbol{\Pi}(n-1, n, n/2)$ formata dalle righe di $\boldsymbol{\Pi}$ tolta la prima colonna;

$\boldsymbol{\Pi}(n-1, 2n, n/2-1)$ formata da $\boldsymbol{\Pi}(n-1, n, n/2)$ e dai suoi complementi.

Esempio 6.12. Si consideri il caso $n = 4$. La

$$\boldsymbol{H_4} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & - & 1 & - \\ 1 & 1 & - & - \\ 1 & - & - & 1 \end{pmatrix} \quad \text{porta a} \quad \boldsymbol{B_4} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

come matrice binaria, con la sostituzione $1 \to 0$ e $-1 \to 1$. Da $\boldsymbol{B_4}$ si ottengono i seguenti tre codici

$C(3,4,2)$	$C(3,8,1)$	$C(4,8,2)$
$\boldsymbol{\mathcal{A}_4} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	$\boldsymbol{\mathcal{B}_4} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\boldsymbol{\mathcal{C}_4} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$

○

Esempio 6.13. Un esempio concreto è legato alle missioni spaziali su Marte *Mariner*, del 1969, nelle quali venne usato un codice non lineare (32,64,16) basato su

$$\boldsymbol{\mathcal{C}_{16}} = \begin{pmatrix} \boldsymbol{H_{32}} \\ \\ \boldsymbol{-H_{32}} \end{pmatrix}$$

Le 64 parole di codice di lunghezza 32 vennero associate a tutte le possibili 6-ple binarie, e ciò consentì di fornire i 64 livelli di grigio nei *pixel* delle immagini trasmesse. Il tasso è pari a 6/32 e si possono correggere fino a 7 errori. Il codice si può ricavare anche dai laterali di un codice di Reed-Muller (32,64) $\mathfrak{R}(1,5)$, e questo fatto consente di individuare metodi efficienti per la codifica/decodifica ◯

I codici di Hadamard sono ad elevata capacità di correzione, ma hanno un tasso asintoticamente nullo. Infatti

$$R_{_{Had}} = \frac{\log 2n}{n} \longrightarrow 0 \qquad \lambda_{_{Had}} = \frac{n/2}{n} = \frac{1}{2} \tag{6.48}$$

6.3.6 Codici perfetti e il codice di Golay

Si è già toccato il problema della perfezione di un codice, cioè la capacità di correggere *tutte e sole* le configurazioni fino a t errori. Dal punto di vista geometrico ciò significa che l'unione di tutte le sfere di raggio t, fra loro prive d'intersezioni, esaurisce $\boldsymbol{q^n}$, cioè che il prodotto del numero delle sfere per il volume di ciascuna di esse eguaglia q^n. La condizione si esprime analiticamente mediante l'equazione (6.16) di *impacchettamento delle sfere*, che riscriviamo per convenienza

$$M\left[\sum_{i=0}^{t}\binom{n}{i}(q-1)^i\right] = q^n \tag{6.49}$$

Si tratta allora di indagare sull'esistenza di quaterne d'interi (n, M, d_{min}, q) che soddisfino l'equazione Diafontea (6.49), e di verificare se, una volta trovata $(n^*, M^*, d^*_{min}, q^*)$ soluzione intera dell'equazione, esiste effettivamente un codice lineare o non lineare con questi parametri.

Oss. *6.6*. La validità dell'equazione (6.49) per un certa quaterna assegnata *non* garantisce l'esistenza di un codice con tali parametri; ciò può essere comprensibile nel caso lineare, visto che la linearità costituisce un vincolo suppletivo (si veda come esempio il teorema 6.4), ma risulta meno intuitivo nel caso non lineare, dove c'è libertà di scelta per le parole di codice. In questo caso diventa determinante il fatto che $\mathbf{0}$ *deve* essere comunque una parola di codice o per il codice in esame o per un suo equivalente che abbia gli stessi parametri.

Si può verificare facilmente che tanto i codici a ripetizione $C(2t+1,1)$ $(q=2)$, quanto i codici di Hamming $C(\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1}-m)$ $(m \geq 2)$, che hanno $d_{min}=3$, soddisfano entrambi la condizione di impacchettamento delle sfere. Per i codici a ripetizione con $n=2t+1$ si ha l'uguaglianza

$$2\left(\sum_{i=0}^{t}\binom{n}{i}\right)=2^n$$

banalmente verificata, mentre per i codici di Hamming possiamo scrivere

$$\begin{cases} q^{n-k}=q^m \\ \binom{n}{0}+\binom{n}{1}(q-1)=q^m \end{cases} \Longrightarrow \binom{n}{0}+\binom{n}{1}(q-1)=q^{n-k}$$

che soddisfa la (6.49).

Nel 1949 Marcel Golay, sulla base di una ricerca praticamente manuale, riuscì a individuare altre tre quaterne, due riferite all'alfabeto binario e una a quello ternario. Esse sono

$$q=2 \qquad (23,2^{12},7) \quad (90,2^{78},5)$$

$$q=3 \qquad (11,3^6,5)$$

Come vedremo nel seguito la $(90,2^{78},5)$ *non porta ad alcun codice*. Per curiosità riferiamo che la terna $(11,3^6,5)$ fu individuata in ben altro contesto due anni prima della sua pubblicazione ufficiale a cura di Golay. Infatti nel 1947, sui numeri 27, 28 e 33 del periodico calcistico finlandese "Veikaaja", comparve una serie di articoli di tale Virtakallio, nei quali la terna in questione era in qualche modo associata agli esiti delle partite di calcio (come noto l'esito è ternario: vittoria, pareggio sconfitta). Anche se tra il 1949 e il 1950 (pubblicazione dell'articolo sui codici di Hamming) furono scoperte molte soluzioni alla (6.49), è straordinario che esse siano rimaste essenzialmente *le sole* disponibili a tutt'oggi. Mediante verifiche sistematiche effettuate al calcolatore, già negli anni '70 si appurò che per

$$n \leq 1000 \quad t \leq 1000 \quad q \leq 100$$

non esistono codici perfetti oltre ai seguenti:

1. il codice *completo* $(n,M,d_{min})=(n,q^n,1)$ formato dallo spazio q^n ;
2. il codice *banale* $(n,1,2n+1)$ formato dalla sola parola nulla;
3. la classe dei codici a *ripetizione* (2t+1,2,2t+1) con $q=2$;

4. la classe dei codici di *Hamming* $(\frac{q^m-1}{q-1}, q^{n-m}, 3)$ $(m \geq 2)$;

5. il codice di *Golay binario* $(23, 2^{12}, 7)$ che corregge 3 errori;

6. il codice *ternario di Golay* $(11, 3^6, 5)$ che corregge 2 errori.

Si noti che, a parte i codici banali, i codici a ripetizione e quelli di Hamming costituiscono una *classe di soluzioni*, mentre entrambi i codici di Golay sono *isolati*. Per lungo tempo si congetturò che i codici di Hamming e di Golay fossero gli unici non banali con i parametri assegnati, ma nel 1962 Vasil'ev individuò dei codici non lineari perfetti con gli stessi parametri di quello di Hamming binario. Successivamente furono scoperti codici non lineari con gli stessi parametri di quelli di Hamming anche su $GF(q)$, con q potenza di un primo. Tuttavia, tra il 1973 e il 1975, Tietäväinen e van Lint dimostrarono definitivamente che *qualunque codice q-ario perfetto non banale, con q potenza di un primo, ha gli stessi parametri dei codici di Hamming o di Golay*. Mentre per i codici di Hamming esistono codici non lineari con gli stessi parametri, ma a questi non equivalenti, ciò fu escluso per il codice di Golay, che rimane unico strutturalmente.

Molti problemi rimangono tuttavia ancora aperti, soprattutto con riferimento al caso in cui q non sia una potenza di un primo.

A completamento dell'analisi fatta dimostriamo il seguente teorema

Teorema 6.4. *Non esiste alcun codice lineare associato alla terna* $(90, 2^{78}, 5)$.

Dim. Sia per assurdo $\boldsymbol{H}$ una matrice di controllo per tale codice. La sua dimensione è 12×90 e dispone dunque di 90 colonne $\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_{90}}$. Poiché $d_{min} = 5$, se prendiamo 4 colonne queste sono linearmente indipendenti. Consideriamo ora l'insieme

$$\boldsymbol{\mathfrak{X}} = \{\boldsymbol{0}, \boldsymbol{h_i}, \boldsymbol{h_j} + \boldsymbol{h_k} : 1 \leq i \leq 90, 1 \leq j < k \leq 90\}$$

Esso contiene $1 + 90 + \binom{90}{2} = 2^{12}$ vettori distinti, e quindi $\boldsymbol{\mathfrak{X}}$ coincide con l'insieme di tutte le 2^{12} n-ple binarie di lunghezza 12. Tra queste metà hanno peso pari e metà dispari. Calcoliamo ora il numero di vettori dispari in altro modo. Supponiamo che m colonne di $\boldsymbol{H}$ abbiano peso dispari, cioè $90 - m$ hanno peso pari. Ricordando la (6.37) possiamo scrivere

$$wt(\boldsymbol{h_j} \oplus \boldsymbol{h_k}) = wt(\boldsymbol{h_j}) + wt(\boldsymbol{h_k}) - 2wt(\boldsymbol{h_j} \cdot \boldsymbol{h_k})$$

e succede che $wt(\boldsymbol{h_j} \oplus \boldsymbol{h_k})$ è dispari se e solo se $wt(\boldsymbol{h_j})$ è dispari con $wt(\boldsymbol{h_k})$ pari o viceversa. Così, un'altra espressione per i vettori di peso dispari di $\boldsymbol{\mathfrak{X}}$ è $m + m(90 - m)$, cioè $m(91 - m) = 2^{11}$, che non può essere soddisfatta per alcun valore di m, poiché tanto m che $91 - m$ dovrebbero essere potenze di 2. □

I codici isolati di Golay $(23, 2^{12}, 7)$ **e** $(11, 3^6, 5)$

Facciamo brevemente cenno alla struttura dei codici isolati di Golay, che costituiscono un vero e proprio punto di singolarità nel contesto delle soluzione relative all'equazione (6.49) d'impacchettamento delle sfere. Punto di partenza in entrambi i casi è il teorema 6.1, che assicura l'esistenza di un codice $(n+1, M, d_{min}+1)$ assegnato che sia un codice (n, M, d_{min}) con d_{min} dispari. Nel nostro caso, accanto al codice perfetto $(23, 2^{12}, 7)$ esiste sicuramente anche un codice (non perfetto) $(24, 2^{12}, 8)$, noto col nome di *codice binario esteso di Golay*. Partiamo dall'illustrazione di quest'ultimo. La sua matrice generatrice 12×24 è la seguente

$$\mathcal{G}_{24} = \left(\begin{array}{c|cccccccccccc|cccccccccccc} 1 & 1 & & & & & & & & & & & & 1 & 1 & & 1 & 1 & 1 & & & & 1 & \\ 1 & & 1 & & & & & & & & & & & & 1 & 1 & & 1 & 1 & 1 & & & & 1 \\ 1 & & & 1 & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 & & & \\ 1 & & & & 1 & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 & & \\ 1 & & & & & 1 & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 & \\ 1 & & & & & & 1 & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 \\ 1 & & & & & & & 1 & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 \\ 1 & & & & & & & & 1 & & & & & & & & & & 1 & & 1 & 1 & & 1 \\ 1 & & & & & & & & & 1 & & & & & & & & & & 1 & & 1 & 1 & \\ 1 & & & & & & & & & & 1 & & & & & & & & & & 1 & & 1 & 1 \\ 1 & & & & & & & & & & & 1 & & & & & & & & & & 1 & & 1 \\ \hline & & & & & & & & & & & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right)$$

dove la matrice 11×11 sulla destra è ricavata da una matrice di Hadamard nella costruzione secondo Paley e la somma di due sue righe ha sempre peso 6. Di conseguenza la somma di ogni coppia di righe di $\mathcal{G}_{24}$ ha peso 8. Si potrebbe inoltre dimostrare che non ci sono parole con peso inferiore e che pertanto $d_{min} = 8$; il codice corregge dunque 3 errori.

Il codice $\mathcal{G}_{23}(23, 2^{12}, 7)$ si costruisce a partire da $\mathcal{G}_{24}$, elidendo una sua qualsiasi colonna. Esso soddisfa l'equazione di impacchettamento delle sfere

$$2^{12}\left[1 + 23 + \binom{23}{2} + \binom{23}{3}\right] = 2^{23}$$

ed è dunque perfetto.

Per quanto riguarda il codice ternario si può procedere in modo simile al caso binario. In questo caso il codice esteso $\mathcal{G}_{12}(12, 3^6, 6)$ ha la seguente matrice generatrice

$$\mathcal{G}_{12} = \left(\begin{array}{cccccc|cccccc} 1 & & & & & & 0 & 1 & 1 & 1 & 1 & 1 \\ & 1 & & & & & 1 & 0 & 1 & 2 & 2 & 1 \\ & & 1 & & & & 1 & 1 & 0 & 1 & 2 & 2 \\ & & & 1 & & & 1 & 2 & 1 & 0 & 1 & 2 \\ & & & & 1 & & 1 & 2 & 2 & 1 & 0 & 1 \\ & & & & & 1 & 1 & 1 & 2 & 2 & 1 & 0 \end{array}\right)$$

e diventa un $\mathcal{G}_{11}(11, 3^6, 5)$ elidendo una colonna qualunque. Anche in questo caso vale la condizione di impacchettamento delle sfere

$$3^6 \left[1 + 2 \cdot 11 + 2^2 \binom{11}{2}\right] = 3^{11}$$

e anche $\mathcal{G}_{11}$ è perfetto.

6.3.7 Codici di Reed-Muller

I codici di Reed-Muller risalgono al 1954 e per quanto le loro prestazioni non siano in generale eccelse, hanno l'indubbio vantaggio di una attuazione molto efficiente delle operazioni di codifica/decodifica; per tale motivo sono stati impiegati con profitto nelle missioni spaziali Mariner, in congiunzione col codice (32,64,16) di Hadamard. Un altro vantaggio, minore, riguarda possibilità di gestire due parametri indipendenti nella loro costruzione, il che offre una maggiore flessibilità nel progetto del codice in termini di tasso rapportato alla capacità di correzione.

Premettiamo alla loro definizione alcune nozioni riguardanti le funzioni Booleane. Si consideri il vettore $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \mathbf{2^m}$ composto da m variabili binarie; ogni applicazione $B(x_1, x_2, \ldots, x_m)$

$$B : \mathbf{2^m} \longrightarrow GF(2)$$

si definisce *funzione Booleana*. Poiché ci sono in tutto $n = 2^m$ posibili vettori, la funzione dovrà specificare il suo valore per ciascuno di essi tramite la cosiddetta *tavola di verità*. Chiamiamo allora $\mathfrak{B}_{\mathfrak{m}}$ l'insieme di tutte le possibili 2^{2^m} funzioni Booleane.

Esempio 6.14. Per $m = 2$ indichiamo le seguenti ben note funzioni AND, OR e XOR

x_1x_2	AND
00	0
01	0
10	0
11	1

x_1x_2	OR
00	0
01	1
10	1
11	1

x_1x_2	XOR
00	0
01	1
10	1
11	0

Per $m = 3$ una delle possibili $2^{2^3} = 256$ funzioni è la seguente

$x_1x_2x_3$	$B(x_1x_2x_3)$
000	0
001	1
010	1
011	0
100	0
101	0
110	1
111	1

○

Ogni funzione Booleana può essere espressa tramite la sua *forma normale disgiuntiva* come unione logica di disgiunzioni tra le variabili x_i (o le loro negazioni $\bar{x}_i$)

$$B(x_1x_2\ldots x_m) = \bigvee_{(a_1a_2\ldots a_m)\in \mathbf{2^m}} b(a_1a_2\ldots a_m)y_1(a_1)y_2(a_2)\ldots y_m(a_m)$$

$$\text{con} \quad y_i(a_i) = \begin{cases} x_i & \text{se} \quad a_i = 1 \\ 1 + x_i & \text{se} \quad a_i = 0 \end{cases} \tag{6.50}$$

cioè y_i assume il valore di x_i oppure della sua negazione $\bar{x}_i = 1 + x_i$. L'unione logica, inoltre, è estesa a tutte la 2^m m-ple binarie e corrisponde a effettuare un "OR" su tutte le m-ple. Il coefficiente $b(a_1a_2\ldots a_m)$ viene invece determinato dalla funzione. Nell'esempio precedente si ottiene l'espressione

$$B(x_1x_2x_3) = \bar{x}_1\bar{x}_2x_3 \ \vee \ \bar{x}_1x_2\bar{x}_3 \ \vee \ x_1x_2\bar{x}_3 \ \vee \ x_1x_2x_3$$

che vale 1 in corrispondenza delle terne 001, 010, 110 e 111.

Poiché ciascuna funzione $B(x_1, x_2, \ldots, x_m) \in \mathfrak{B}_\mathfrak{m}$ è associata a un vettore di lunghezza 2^m, possiamo introdurre una somma tra funzioni strutturando $\mathfrak{B}_\mathfrak{m}$ come spazio vettoriale su $GF(2)$ di dimensione 2^m. Ciò consente di esprimere ogni funzione Booleana mediante una base opportuna dello spazio, introducendo quindi una rappresentazione in *forma algebrica* alternativa alla forma normale disgiuntiva. Nel seguito considereremo una base particolare che porta a una rappresentazione di notevole interesse.

Cominciamo col considerare ogni funzione come vettore binario di lunghezza 2^m e di esprimerla mediante la base canonica

$$\boldsymbol{c_i} = (0\ldots010\ldots0)$$

con 1 in posizione i ($1 \leq i \leq 2^m$). Il legame tra forma disgiuntiva e forma algebrica viene messo in luce interpretando gli elementi della base canonica $\boldsymbol{c_1, c_2, \ldots, c_{2^m}}$ come funzioni, e sostituendo i vettori (10 ... 00), (01 ... 00), ... , (00 ... 01) nella forma normale disgiuntiva (6.50). Ponendo tutte le m-ple del dominio in ordine lessicografico si ottengono le funzioni

$$\begin{array}{c}(1+x_1)(1+x_2)\ldots(1+x_{m-1})(1+x_m)\\(1+x_1)(1+x_2)\ldots(1+x_{m-1})x_m\\\vdots\\(1+x_1)x_2\ldots x_{m-1}x_m\\\vdots\\x_1x_2\ldots x_{m-1}(1+x_m)\\x_1x_2\ldots x_{m-1}x_m\end{array} \tag{6.51}$$

Si consideri la prima tra queste; essa può essere riscritta nel modo seguente

$$\begin{aligned}\prod_{i=1}^{m}(1+x_i) = 1 &+ x_1 + x_2 + \ldots + x_m +\\ &+ x_1x_2 + x_1x_3 + \ldots + x_{m-1}x_m +\\ &\vdots\\ &+ x_1x_2\ldots x_{m-1}x_m\end{aligned} \tag{6.52}$$

Inoltre, ogni altra funzione della (6.51) si ottiene considerando un sottinsieme delle somme che compaiono nella (6.52). Come esempio consideriamo il caso $m = 3$

$$\begin{array}{ll}000 & (1+x_1)(1+x_2)(1+x_3) = 1+x_1+x_2+x_3+x_1x_2+x_1x_3+x_2x_3+\\001 & (1+x_1)(1+x_2)x_3 = x_3+x_1x_3+x_2x_3+x_1x_2x_3 \qquad\qquad +x_1x_2x_3\\010 & (1+x_1)x_2(1+x_3) = x_2+x_1x_2+x_2x_3+x_1x_2x_3\\011 & (1+x_1)x_2x_3 = x_2x_3+x_1x_2x_3\\100 & x_1(1+x_2)(1+x_3) = x_1+x_1x_3+x_1x_2+x_1x_2x_3\\101 & x_1(1+x_2)x_3 = x_1x_3+x_1x_2x_3\\110 & x_1x_2(1+x_3) = x_1x_2+x_1x_2x_3\\111 & x_1x_2x_3\end{array}$$

Ogni elemento della base canonica $\boldsymbol{c_i}$ si ricava dunque da una opportuna combinazione lineare degli elementi che compongono la (6.52); poiché un qualunque vettore di $\boldsymbol{2^m}$, cioè una qualunque funzione Booleana $B(x_1x_2\ldots x_m)$, si ottiene da un'opportuna combinazione lineare degli elementi della base canonica $\boldsymbol{c_1, c_2, \ldots, c_{2^m}}$, lo stesso vettore è esprimibile nella seguente *forma normale*

algebrica

$$\begin{aligned} B(x_1x_2\ldots x_m) = a_0 &+ a_1x_1 + a_2x_2 + \ldots + a_mx_m + \\ &+ a_{1,2}x_1x_2 + a_{1,3}x_1x_3 + \ldots + a_{m-1,m}x_{m-1}x_m + \\ &\vdots \\ &+ a_{1,2,\ldots,m}x_1x_2\ldots x_{m-1}x_m \end{aligned}$$

e dunque i vettori

$$\mathbf{1}, \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}, \boldsymbol{x_1x_2}, \boldsymbol{x_1x_3}, \ldots, \boldsymbol{x_{m-1}x_m}, \ldots, \boldsymbol{x_1x_2\ldots x_m} \qquad (6.53)$$

costituiscono una base $\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_{2^m}}$ dello spazio, con $\mathbf{1} = (1, 1, \ldots, 1)$. Nella tabella sotto riportata si considera il caso $m = 3$

	0	1	2	3	4	5	6	7
$\mathbf{1}$	1	1	1	1	1	1	1	1
$\boldsymbol{x_1}$					1	1	1	1
$\boldsymbol{x_2}$			1	1			1	1
$\boldsymbol{x_3}$		1		1		1		1
$\boldsymbol{x_1x_2}$							1	1
$\boldsymbol{x_1x_3}$						1		1
$\boldsymbol{x_2x_3}$				1				1
$\boldsymbol{x_1x_2x_3}$								1

In generale un codice di Reed-Muller è costituito da un opportuno *sottoinsieme di funzioni Booleane*.

Def. 6.7. *Un codice binario di Reed-Muller $\mathfrak{R}(r, m)$ di ordine r di lunghezza $n = 2^m$, con $0 \leq r \leq m$, è costituito dalla combinazione lineare di tutti i vettori della base (6.53) fino al prodotto di tutte le $\binom{m}{r}$ configurazioni $\boldsymbol{x_{i_1}x_{i_2}\ldots x_{i_r}}$ con $1 \leq i_j \leq m$.*

Esempio 6.15. Si riconsideri il caso $m = 3$ visto precedentemente, con $r = 1$; in tal modo si costruisce il codice di Reed-Muller del primo ordine $\mathfrak{R}(1, 3)$, con $n = 2^3 = 8$ come lunghezza della parole di codice. La matrice generatrice è formata dal vettore $\mathbf{1}$ e dai singoli vettori $\boldsymbol{x_1x_2x_3}$

$$\begin{matrix} \mathbf{1} \\ \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \end{matrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Si nota subito che la matrice è quella duale di un codice di Hamming esteso.◯

Dalla definizione risulta chiaro che i parametri del codice sono

$$k = \binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{r} \qquad n = 2^m \tag{6.54}$$

Per quanto riguarda la capacità di correzione del codice, e quindi la distanza minima, conviene dare qualche ragguaglio sulle tecniche di codifica/decodifica del codice. Per la codifica non ci sono particolari osservazioni; se $\boldsymbol{u} = (u_0 u_1 u_2 \ldots \ldots u_m u_{1,2} u_{1,3} \ldots u_{m-1,m} \ldots u_{1,\ldots,r} \ldots u_{m-r+1,\ldots,m})$ è il vettore d'ingresso di lunghezza k, detto $\boldsymbol{x} = (x_1 x_2 \ldots x_n)$ il vettore relativo alla parola di codice si ha semplicemente

$$\begin{aligned}\boldsymbol{x} = \boldsymbol{uG} = u_0\mathbf{1} &+ u_1\boldsymbol{x_1} + u_2\boldsymbol{x_2} + \ldots + u_m\boldsymbol{x_m} + \\ &+ u_{1,2}\boldsymbol{x_1 x_2} + u_{1,3}\boldsymbol{x_1 x_3} + \ldots + u_{m-1,m}\boldsymbol{x_{m-1} x_m} + \\ &+ \ldots + \\ &+ u_{1,\ldots,r}\boldsymbol{x_1 \ldots x_r} + \ldots + u_{m-r+1,\ldots,m}\boldsymbol{x_{m-r+1} \ldots x_m}\end{aligned}$$

Per quanto riguarda la decodifica si può notare che, per ogni coordinata $u_{i_1 \ldots i_h}$ $1 \leq h \leq r$ (compresa la u_0) del vettore $\boldsymbol{u}$ d'informazione, esistono un certo numero di *equazioni di controllo* indipendenti le quali, in assenza di errori, forniscono come risultato sempre $u_{i_1,\ldots i_h}$. Se facciamo riferimento al caso $m = 4$ della tabella sotto, con $r = 2$, si ottiene la tabella del codice $\mathfrak{R}(2,4)$ che costituisce la matrice generatrice di un C(16,11)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\boldsymbol{x_1}$									1	1	1	1	1	1	1	1
$\boldsymbol{x_2}$					1	1	1	1					1	1	1	1
$\boldsymbol{x_3}$			1	1			1	1			1	1			1	1
$\boldsymbol{x_4}$		1		1		1		1		1		1		1		1
$\boldsymbol{x_1 x_2}$													1	1	1	1
$\boldsymbol{x_1 x_3}$											1	1			1	1
$\boldsymbol{x_1 x_4}$										1		1		1		1
$\boldsymbol{x_2 x_3}$							1	1							1	1
$\boldsymbol{x_2 x_4}$						1		1						1		1
$\boldsymbol{x_3 x_4}$				1				1				1				1

Dalla tabella si può notare che la somma delle prime 4 colonne della stessa fornisce valore zero in tutte le componenti, tranne che in quella relativa al termine $u_{3,4}$ dell'ultima riga, in corrispondenza della zona tratteggiata. Ciò comporta che, per la somma delle prime 4 componenti di $\boldsymbol{x}$, sia $u_{3,4} = x_1 + x_2 + x_3 + x_4$. Lo

stesso accade però per le colonne 5-6-7-8, 9-10-11-12 e 13-14,15,16. Possiamo allora scrivere le seguenti 4 equazioni di controllo

$$u_{3,4} = x_1 + x_2 + x_3 + x_4 = x_5 + x_6 + x_7 + x_8 = \\ = x_9 + x_{10} + x_{11} + x_{12} = x_{13} + x_{14} + x_{15} + x_{16}$$

la prima delle quali è associata al tratteggio delle posiizoni 1,2,5,6 in figura. In generale le equazioni di controllo sono 2^{m-r}, e se si verificano degli errori qualcuna di queste non sarà più soddisfatta. Per effetto delle struttura della matrice il procedimento può essere esteso a tutte le componenti del vettore $\boldsymbol{u}$, e le colonne da considerare nella prima somma sono quelle coperte delle aree tratteggiate in corrispondenza di una riga. Per esempio, nel caso della penultima riga si ha

$$u_{2,4} = x_1 + x_2 + x_5 + x_6 = x_3 + x_4 + x_7 + x_8 = \\ = x_9 + x_{10} + x_{13} + x_{14} = x_{11} + x_{12} + x_{15} + x_{16}$$

La prime equazioni per le altre componenti sono

$$u_{1,2} = x_1 + x_5 + x_9 + x_{13} = \dots \\ u_{1,3} = x_1 + x_3 + x_9 + x_{11} = \dots \\ u_{1,4} = x_1 + x_2 + x_9 + x_{10} = \dots \\ u_{2,3} = x_1 + x_3 + x_5 + x_7 = \dots$$

Per le componenti rimanenti vale un discorso analogo, con la sola differenza che il numero di equazioni di controllo raddoppia. Per ricavare le equazioni relative alla componente u_4 si deve considera che la somma delle prime due colonne ha un'unica componente unitaria in corrispondenza della riga associata a u_4, e così abbiamo le seguenti 8 equazioni per ciascuna componente

$$u_4 = x_1 + x_2 = x_3 + x_4 = x_5 + x_6 = x_7 + x_8 = \\ = x_9 + x_{10} = x_{11} + x_{12} = x_{13} + x_{14} = x_{15} + x_{16} \\ u_3 = x_1 + x_3 = x_2 + x_4 \dots \\ u_2 = x_1 + x_5 = x_2 + x_6 \dots \\ u_1 = x_1 + x_9 = x_1 + x_{10} \dots$$

Per l'ultima componente abbiamo 16 equazioni di controllo, una per ciascuna componente. È allora evidente che la decisione sul valore della cifra d'informazione può essere presa *a maggioranza*, nel senso che si deciderà 0 se la maggioranza delle equazioni di controllo indica 0 come soluzione, 1 altrimenti. Tale procedimento di *decodifica a maggioranza* rappresenta una generalizzazione della decodifica vista per i codici a ripetizione, che sono codici di Reed-Muller di ordine 0. Vediamo un esempio concreto di decodifica per il codice $\mathfrak{R}(2,4)$.

Esempio 6.16. Sia $\boldsymbol{u} = (10010000110)$ il vettore d'informazione. La parola di codice si ottiene sommando la $1^a, 4^a, 9^a$ e 10^a riga della matrice generatrice e corrisponde a

$$\boldsymbol{x} = \mathbf{1} + \boldsymbol{x_3} + \boldsymbol{x_2 x_3} + \boldsymbol{x_2 x_4}$$

$\mathbf{1}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\boldsymbol{x_3}$			1	1			1	1			1	1			1	1
$\boldsymbol{x_2 x_3}$							1	1							1	1
$\boldsymbol{x_2 x_4}$						1		1						1		1
$\boldsymbol{x}$	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0

e dunque $\boldsymbol{x} = (1100101011001010)$. Supponiamo che si sia verificato un errore in posizione 1, e che il vettore ricevuto sia $\boldsymbol{y} = (0100101011001010)$. Le 4 equazioni di controllo per ogni componente con $r = 2$ sono le seguenti

$$\begin{array}{l} 0+1+0+0=1 \\ 1+0+1+0=0 \\ 1+1+0+0=0 \\ 1+0+1+0=0 \end{array} \quad u_{3,4}=0 \qquad \begin{array}{l} 0+0+1+1=0 \\ 1+0+0+0=1 \\ 1+0+1+1=1 \\ 1+0+0+0=1 \end{array} \quad u_{2,3}=1$$

$$\begin{array}{l} 0+1+1+0=0 \\ 0+0+1+0=1 \\ 1+1+1+0=1 \\ 0+0+1+0=1 \end{array} \quad u_{2,4}=1 \qquad \begin{array}{l} 0+1+1+1=1 \\ 0+0+0+0=0 \\ 1+0+1+0=0 \\ 1+0+1+0=0 \end{array} \quad u_{1,4}=0$$

$$\begin{array}{l} 0+0+1+0=1 \\ 1+0+1+0=0 \\ 1+1+1+1=0 \\ 0+0+0+0=0 \end{array} \quad u_{1,3}=0 \qquad \begin{array}{l} 0+1+1+1=1 \\ 1+0+1+0=0 \\ 0+1+0+1=0 \\ 0+0+0+0=0 \end{array} \quad u_{1,2}=0$$

e la decodifica degli u_{ij} avviene a maggioranza. Come si può notare il bit errato in prima posizione ha interferito con tutti i 6 blocchi di equazioni, come si deduce del resto dalla tabella di $\mathfrak{R}(2,4)$. A questo punto possiamo sottrarre da $\boldsymbol{y}$ (sommare su $GF(2)$) il contributo dei vettori corrispondenti ai coefficienti diversi da 0, ottenendo

$$\boldsymbol{y^{(1)}} = \boldsymbol{y} + \boldsymbol{x_2 x_3} + \boldsymbol{x_2 x_4}$$

$\boldsymbol{y}$	0	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0
$\boldsymbol{x_2 x_3}$							1	1							1	1
$\boldsymbol{x_2 x_4}$						1		1						1		1
$\boldsymbol{y^{(1)}}$	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

Le equazioni che consentono di ricavare gli u_i sono le seguenti:

$$\begin{array}{llll} 0+1=1 & 0+0=0 & 1+1=0 & 0+0=0 \\ 1+1=0 & 0+0=0 & 1+1=0 & 0+0=0 \end{array} \qquad u_4 = 0$$

$$\begin{array}{llll} 0+0=0 & 1+0=1 & 0+1=1 & 0+1=1 \\ 1+0=1 & 1+0=1 & 0+1=1 & 0+1=1 \end{array} \qquad u_3 = 1$$

$$\begin{array}{llll} 0+1=1 & 1+1=0 & 0+0=0 & 0+0=0 \\ 1+1=0 & 1+1=0 & 0+0=0 & 0+0=0 \end{array} \qquad u_2 = 0$$

$$\begin{array}{llll} 0+1=1 & 1+1=0 & 0+0=0 & 0+0=0 \\ 1+1=0 & 1+1=0 & 0+0=0 & 0+0=0 \end{array} \qquad u_1 = 0$$

Rimuovendo da $\boldsymbol{y^{(1)}}$ il vettore $\boldsymbol{x_3}$ si ottiene

$\boldsymbol{y^{(1)}}$	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
$\boldsymbol{x_3}$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$\boldsymbol{y^{(2)}}$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

che a maggioranza indica $u_0 = 1$. Il vettore d'informazione viene dunque recuperato senza errori ◯

Per quanto riguarda la distanza minima, essa si può ricavare sulla base del seguente ragionamento: poiché ci sono 2^{m-r} equazioni di controllo, decidendo a maggioranza su queste si possono tollerare $(2^{m-r}/2) - 1 = 2^{m-r-1} - 1$ equazioni sbagliate; ma ogni errore si ripercuote solo su una equazione, e dunque si possono tollerare al più $2^{m-r-1}-1$ errori. Deve essere allora $d_{min} \geq 2t+1 = 2(2^{m-r-1} - 1) + 1 = 2^{m-r} - 1$; ma le parole di codice hanno peso pari, e per effetto della (6.37) anche la distanza minima deve essere pari, cioè $d_{min} = 2^{m-r}$.

Per quanto riguarda le prestazioni asintotiche si ha

$$\lambda_{RM} = \frac{2^{m-r}}{2^m} = \frac{1}{2^r} \qquad R_{RM} = \frac{\sum_{i=0}^{r} \binom{m}{r}}{2^m} \tag{6.55}$$

e dunque se r è costante $\lambda_{RM} > 0$, ma $R_{RM} \to 0$. Ponendo invece $r = \alpha m$ si ottiene $R_{RM} > 0$, ma $\lambda_{RM} \to 0$. In entrambi i casi non si riesce dunque a tener discosti dallo 0 tanto il tasso che la capacità di correzione. Si noti tuttavia che per i codici di Reed-Muller si ha a disposizione un grado di libertà in più nella scelta dei parametri, che consente una maggior flessibilità (al finito) nella scelta del compromesso tra capacità di correzione e tasso.

6.4 Codici ciclici

I codici ciclici costituiscono una sorta di paradigma di codifica, all'interno del quale possiamo trovare tutti i codici lineari finora incontrati (Hamming, BCH, Golay). L'idea di base è quella di sfruttare, fra le tante matrici generatrici equivalenti per un codice assegnato, quella che consente di costruire accanto a ciascuna parola tutte le altre parole di codice che sono ottenute mediante *traslazione ciclica delle componenti*; più precisamente si ha la seguente

Def. 6.8. *Un codice si dice ciclico quando una permutazione ciclica di una sua parola di codice è ancora una parola di codice*

$$\boldsymbol{a} = (a_0 a_1 \ldots a_{n-1}) \in \mathfrak{C} \quad \Longrightarrow \quad \boldsymbol{a'} = (a_{n-1} a_0 \ldots a_{n-2}) \in \mathfrak{C}$$

In tal caso possiamo parlare di codice costituito da un *sottospazio lineare ciclico* dello spazio vettoriale ambiente.

L'efficacia della descrizione dei codici lineari tramite il paradigma dei codici ciclici è relativa al fatto che la composizione delle parole, oltre che obbedire ai vincoli determinati dalla struttura di spazio vettoriale, può essere associata con profitto all'anello $F[x]$ dei polinomi con coefficienti su un campo finito $GF(q)$, il che determina drastiche semplificazioni nel progetto del codice e nell'automazione delle operazioni di codifica/decodifica. In quest'ultimo caso si possono infatti usare in modo sistematico i *registri lineari a scorrimento retroazionato*, che analizzaremo nei dettagli nella sezione 6.5. Associamo a ciascuna parola di codice un polinomio

$$\boldsymbol{a} = (a_0 a_1 \ldots a_{n-1}) \quad \Longrightarrow \quad a(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

secondo quanto già visto precedentemente (si veda p.es. l'associazione fatta nella sezione 6.3.4 tra colonne della matrice di controllo e polinomi); usiamo per semplicità la notazione $a(x) \in \mathfrak{C}$ per indicare che $a(x)$ è il polinomio associato alla parola di codice $\boldsymbol{a} \in \mathfrak{C}$. Eseguiremo le somme tra polinomi seguendo le regole usuali, mentre le moltiplicazioni vanno fatte sfruttando una riduzione $\bmod d(x)$, con $d(x)$ polinomio opportuno. Nel nostro caso scegliamo

$$d(x) = x^n - 1$$

La riduzione $\bmod (x^n - 1)$ ci fa passare dall'anello dei polinomi $F[x]$ all'anello quoziente $R_n = F[x]/(x^n - 1)$ delle *classi di resto* $\bmod (x^n - 1)$, dove $x^n - 1 \equiv 0$. Si noti che effettuare una permutazione ciclica degli elementi $(a_0 a_1 \ldots a_{n-1})$ nell'anello quoziente equivale a una moltiplicazione del polinomio $a(x)$ per x

$$\begin{aligned} x\, a(x) &= x(a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}) = \\ &= a_0 x + a_1 x^2 + \ldots + a_{n-1}(x^n - 1) + a_{n-1} \\ &\equiv a_{n-1} + a_0 x + \ldots + a_{n-2} x^{n-1} \end{aligned}$$

Dalla sezione 6.2.1 ricordiamo inoltre che un sottoanello $\boldsymbol{I}$ di un anello $\boldsymbol{R}$ si dice *ideale* di $\boldsymbol{R}$ se, per ogni $\boldsymbol{a} \in \boldsymbol{I}$ e $\boldsymbol{r} \in \boldsymbol{R}$ si ha $\boldsymbol{r} \cdot \boldsymbol{a} \in \boldsymbol{I}$ e $\boldsymbol{a} \cdot \boldsymbol{r} \in \boldsymbol{I}$. Vale allora il seguente teorema, che caratterizza la struttura dei codici ciclici

Teorema 6.5. *Un codice è ciclico se e solo se esso è un ideale nella classe dei resti* $\mod (x^n - 1)$.

Dim. Se $\boldsymbol{a}$ è una qualunque parola di un codice ciclico $\mathfrak{C}$, sia $a(x)$ il polinomio associato. Dunque se $a(x) \in \mathfrak{C}$ anche $x\ a(x) \in \mathfrak{C}$ in quanto ciclico. Si deve dimostrare che $b(x)\ a(x) \in \mathfrak{C}\ \forall\ b(x)$. Sia $b(x) = b_0 + b_1 x + \ldots + b_k x^k$. Il prodotto vale

$$b_0 a(x) + b_1 x a(x) + b_2 x^2 a(x) + \ldots + b_k x^k a(x)$$

Ma $b_0 a(x) \in \mathfrak{C}$ e anche $b_1 x a(x)$, in quanto il codice è ciclico; segue dunque $b_2 x(x a(x)) \in \mathfrak{C}$ in quanto $x a(x) \in \mathfrak{C}$ e così via; cioè il codice è un ideale.

Viceversa sia $\mathfrak{C}$ un ideale e prendiamo $a(x) \in \mathfrak{C}$. Dal fatto che è un ideale sappiamo anche che $x a(x) \in \mathfrak{C}$, e dunque il codice è ciclico. □

Il codice ciclico è dunque un sottoanello con la *proprietà dell'assorbimento* . L'analogia cui si ricorre solitamente è quella dell'anello degli interi e del suo sottoanello costituito dai *multipli* di un certo intero m. Anche l'anello in questione, come vedremo fra poco, è formato da un polinomio $m(x)$ e dai suoi multipli. Consideriamo infatti il polinomio monico $m(x) \in \mathfrak{C}$ (cioè col coefficiente di grado più elevato pari a 1) di grado minimo f.

Teorema 6.6. *Nell'anello di un codice ciclico* $\mathfrak{C}$ *di* R_n *accade che*

1. *il polinomio monico* $m(x) \in \mathfrak{C}$ *di grado minimo* f *è unico;*
2. *ogni polinomio* $a(x) \in \mathfrak{C}$ *è multiplo di* $m(x)$.

Dim. Per la 1) si supponga che esistano due polinomi monici di grado minimo

$$m(x) = x^f + m_{f-1} x^{f-1} + \ldots + m_1 x + m_0$$
$$d(x) = x^f + h_{f-1} x^{f-1} + \ldots + h_1 x + h_0$$

Facendo la loro differenza si otterrebbe un polinomio di grado $< f$, contro l'ipotesi che $m(x)$ sia di grado minimo. Deve essere allora $m(x) = d(x)$.

Per la 2) Si prenda un qualunque $a(x) \in \mathfrak{C}$ e lo si divida per $m(x)$

$$a(x) = q(x) m(x) + r(x) \quad \text{con} \quad \text{gr}[r(x)] < f$$

Si ha inoltre $r(x) = a(x) - q(x) m(x) \in \mathfrak{C}$, in quanto tanto $a(x)$ che $q(x) m(x)$ stanno nell'ideale, ed essendo quest'ultimo un sottoanello anche la loro differenza è un elemento dell'ideale. Si noti però che $\text{gr}[r(x)] < \text{gr}[m(x)]$, il che

implica $r(x) \equiv 0$. $\square$

In un ideale esiste dunque un polinomio monico minimo $m(x)$, chiamato *generatore*, e tutti gli altri polinomi dell'ideale si ottengono moltiplicando $m(x)$ per un qualunque altro polinomio.

Per caratterizzare un codice ciclico tramite la matrice generatrice basta allora fornire la sua parola $g(x)$, costituita dal polinomio minimo, e le successive traslazioni $xg(x), x^2g(x), x^3g(x), \ldots$, che corrispondono evidentemente a righe linearmente indipendenti. Se poniamo $k = n - f$ e $m(x) = g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}$ si ha la seguente matrice $k \times n$ generatrice del codice

$$\boldsymbol{G} = \begin{pmatrix} g(x) & & & \\ & x\,g(x) & & \\ & & \ddots & \\ & & & x^{k-1}\,g(x) \end{pmatrix} =$$

$$= \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & & & \\ & g_0 & g_1 & \cdots & g_{n-k} & & \\ & & \ddots & & & \ddots & \\ & & & g_0 & g_1 & \cdots & g_{n-k} \end{pmatrix}$$

Un altro metodo per costruire la matrice è quello di dividere il polinomio x^i per $g(x)$

$$x^i = q_i(x)g(x) + r_i(x) \qquad i = n-k, n-k+1, \ldots, n-1$$

Si noti infatti che $x^i - r_i(x) = g(x)q_i(x)$ è una parola di codice, e le k ottenibili in questa forma sono linearmente indipendenti. La matrice generatrice indotta si presenta in forma canonica

$$\boldsymbol{G} = (-R \quad I_{n-k})$$

e il codice è sistematico.

Consideriamo ora $x^n - 1$, che sta nella classe dello 0, e dividiamolo per il polinomio minimo

$$x^n - 1 = q(x)g(x) + r(x) \equiv 0$$

e dunque $r(x) \equiv q(x)g(x)$, che non può essere poiché $\mathrm{gr}[r(x)] < \mathrm{gr}[g(x)]$, a meno che non sia $r\,(x) \equiv 0$. Dunque $x^n - 1$ è un multiplo di $g(x)$ e ogni codice ciclico ha un generatore $g(x)$ che è fattore di $x^n - 1$. Assegnato dunque n dobbiamo scomporre $x^n - 1$ nei suoi fattori irriducibili; ogni fattore è un generatore

per un codice ciclico. Se $x^n - 1$ si scompone nel prodotto di t fattori irriducibili distinti

$$x^n - 1 = f_1(x)f_2(x)\dots f_t(x) = g_i(x)h_i(x) \equiv 0 \qquad i = 1, 2, \dots, 2^t$$

vi saranno in tutto 2^t diversi codici ciclici, con $g_i(x)$ e $h_i(x)$ costituiti dal prodotto di alcuni tra i t fattori, in modo che ogni f_j fattorizzi $g_i(x)$ oppure $h_i(x)$ e che tutti i fattori vengano usati. Nel seguito studieremo meglio il problema della fattorizzazione di $x^n - 1$.

La caratterizzazione della matrice di controllo per un codice ciclico parte dalla considerazione che, assegnato $g(x)$ generatore, si ha

$$x^n - 1 = g(x)h(x) \equiv 0$$

e se $a(x) = g(x)b(x)$ è una parola di codice, avendo $g(x)$ come fattore si ottiene

$$h(x)a(x) = h(x)a(x) = h(x)g(x)b(x) \equiv 0$$

L'equazione $h(x)a(x) \equiv 0$ funge dunque da *equazione di controllo del codice* e $h(x)$ viene definito *polinomio di controllo*; d'altra parte essa non è l'unica, poiché valgono anche le seguenti $n - k - 1$ uguaglianze

$$\begin{aligned} a(x)\ x\ h(x) &\equiv 0 \\ a(x)\ x^2\ h(x) &\equiv 0 \\ a(x)\ x^3\ h(x) &\equiv 0 \\ &\vdots \\ a(x)\ x^{n-k-1}\ h(x) &\equiv 0 \end{aligned}$$

Si noti però che le equazioni (6.56) non possono essere impiegate così come stanno per la costruzione della matrice $\boldsymbol{H}$; esse garantiscono infatti che il prodotto tra i due polinomi $a(x)$ e $x^i\ h(x)$ sta nella classe dello 0; per la costruzione della matrice di controllo ci si viceversa assicurare che il *prodotto scalare tra la parola e il vettore sia uguale a zero*. Ciò può essere interpretato anche dicendo che $h(x)$ non è necessariamente un generatore del codice duale $\mathfrak{C}^\perp$. Tuttavia il polinomio generatore di $\mathfrak{C}^\perp$ può essere espresso in funzione di $h(x)$, come specifica la seguente

Proposizione 6.3. *Siano* $a(x) = (a_0, a_1, \dots, a_{n-1})$ *e* $b(x) = (b_0 b_1 \dots b_{n-1})$. *Allora* $a(x)b(x) \equiv 0$ *se e solo se* $a(x)$ *è ortogonale al vettore* $(b_{n-1} \dots b_1 b_0)$ *e a tutte le sue traslazioni cicliche*.

Dim. Eseguendo il prodotto e riducendo $\mod (x^n - 1)$ si ottiene

$$\begin{aligned} a(x)b(x) = &(a_0 b_{n-1} + a_1 b_{n-2} \ldots + a_{n-1} b_0) x^{n-1} + \\ &+ (a_0 b_0 + a_1 b_{n-1} \ldots + a_{n-1} b_1) x + \\ &+ (a_0 b_1 + a_1 b_0 \ldots + a_{n-1} b_2) x^2 + \\ &\qquad \vdots \\ &+ (a_0 b_{n-2} + a_1 b_{n-3} \ldots + a_{n-1} b_{n-1}) x^{n-2} \end{aligned}$$

e uguagliando a zero tutti i coefficienti si giunge alla tesi □

Il polinomio generatore di $\mathfrak{C}^{\perp}$ è allora il *reciproco* di $h(x)$, cioè $h^*(x) = x^k \cdot h(1/x)$. La matrice di controllo è data da

$$\boldsymbol{H} = \begin{pmatrix} & & & \overleftarrow{h(x)} \\ & & x\,\overleftarrow{h(x)} & \\ & \iddots & & \\ x^{n-k-1}\,\overleftarrow{h(x)} & & & \end{pmatrix} =$$

$$= \begin{pmatrix} & & h_k & \ldots & h_1 & h_0 \\ & h_k & \ldots & h_1 & h_0 & \\ \iddots & & & \iddots & & \\ h_k & \ldots & h_1 & h_0 & & \end{pmatrix} \tag{6.56}$$

dove la freccia a sinistra indica che i coefficienti del polinomio vanno presi crescenti da destra a sinistra.

6.4.1 Fattorizzazione del polinomio $x^n - 1$

La costruzione di un codice ciclico è connessa con la fattorizzazione di $x^n - 1$ su un campo finito $GF(q)$, che in generale non è semplice da effettuare; una trattazione esauriente dell'argomento la si trova sul libro di Berlekamp [9], autore fra l'altro di un algoritmo per la fattorizzazione di un polinomio qualunque.

Tuttavia, lo studio della struttura algebrica dei campi finiti di Galois $GF(p^m)$ (p primo) offre una soluzione molto efficiente connessa con la rappresentazione degli elementi del campo. Purtroppo non possiamo addentrarci in un'analisi sistematica della sua struttura, poiché ciò esula dai contenuti di questo testo. Rimandiamo nuovamente il lettore interessato al testo di Berlekamp [9], che offre un'eccellente trattazione della struttura dei campi finiti, e al testo [69] di Pretzel (in particolare si faccia attenzione alla sezione 12.2); ma si veda anche il testo classico di Cohn [18] oltre al testo in italiano [6] della Berardi .

Nel nostro ambito ci accontenteremo solo di riferire alcuni risultati che ci dànno accesso operativo alla fattorizzazione e alla costruzione dei codici.

Si consideri un campo finito $GF(p^m)$; esso è costituito da p^m elementi, tra cui lo 0 che è l'elemento neutro del gruppo additivo, e l'unità che è l'elemento neutro del gruppo moltiplicativo $GF(p^m) - \{0\}$. Se α è un elemento del campo si possono calcolare le sue successive potenze positive $\alpha, \alpha^2, \alpha^3, \ldots$ o negative $\alpha^{-1}, \alpha^{-2}, \alpha^{-3}, \ldots$. Poichè il campo è finito non tutte le potenze possono essere distinte, e devono esistere esponenti h e k tali che $\alpha^h = \alpha^k$, cioè $\alpha^{h-k} = 1$. Il più piccolo intero d tale che $\alpha^d = 1$ si dice *ordine moltiplicativo di* α. Se $d = p^m - 1$ l'elemento α si dice *primitivo*.

Se consideriamo la struttura additiva del campo, sommando ripetutamente l'unità 1+1+1+ ... , si mettono in evidenza tutti gli elementi *interi* del campo, che non possono essere tutti distinti; deve allora accadere che $\sum_{i=1}^{h} 1 = \sum_{i=1}^{k} 1$, cioè $\sum_{i=1}^{h-k} 1 = 0$. Il più piccolo intero p tale che $\sum_{i=1}^{p} 1 = 0$ si dice *caratteristica* del campo. Si può verificare facilmente che la caratteristica di un campo è un numero primo e che gli interi formano una *parte stabile* rispetto all'addizione e alla moltiplicazione. Di conseguenza $\left(\sum_{i=1}^{h} 1\right) \cdot \left(\sum_{i=1}^{k} 1\right) = \left(\sum_{i=1}^{h \cdot k} 1\right)$. In generale valgono i seguenti importanti risultati, per le dimostrazioni dei quali rimandiamo a [69].

Teorema 6.7.

1. *Se α ha ordine d α^k ha ordine $\frac{d}{(d,k)}$ (con (d,k) MCD tra d e k).*
2. *In ogni campo finito $GF(p^m)$ esiste una radice primitiva di ordine $p^m - 1$ le cui potenze distinte coincidono con tutti gli elementi non nulli del campo.*

Gli elementi del campo sono dunque tutte e sole le soluzioni dell'equazione

$$x^{p^m} - x = x\left(x^{p^m - 1} - 1\right) = 0 \tag{6.57}$$

che corrisponde alle due equazioni $x = 0$, soddisfatta dallo 0, e $x^{p^m - 1} - 1$, soddisfatta dagli elementi non nulli del campo. Ciò suggerisce che una qualunque fattorizzazione di $x^{p^m - 1} - 1 = g(x)h(x)$ corrisponde a una partizione degli elementi non nulli del campo, in quanto ciascun elemento è radice di $g(x)$ *oppure* di $h(x)$. Si noti ora che, a norma del teorema 6.7, tutti gli elementi del campo sono esprimibili come potenza di un elemento primitivo α, e dunque se α ha ordine n gli elementi $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ sono tutte radici di $x^n - 1 = 0$, che peraltro non possiede altre radici. Possiamo allora scrivere

$$x^n - 1 = \prod_{i=0}^{n-1} \left(x - \alpha^i\right) \tag{6.58}$$

Una possibilità è quella di partizionare gli elementi secondo il loro *ordine moltiplicativo*. Si osservi ora che se d è un divisore di n, cioè $n = kd$ (che indichiamo con la notazione d/n), allora $\alpha^k, \alpha^{2k}, \ldots, \alpha^{dk}$ sono tutte radici di $x^d - 1 = 0$, che non ne possiede altre. Le potenze distinte di α contengono dunque tutte le radici d-esime dell'unità. Inoltre, sempre per il teorema 6.7 ogni potenza di α ha un ordine che divide n. Ciò suggerisce di ripartire le potenze di α secondo il loro ordine d divisore di n, giungendo alla seguente fattorizzazione di $x^n - 1$

$$x^n - 1 = \prod_{\substack{d \\ d/n}} \prod_{\substack{\beta \\ \mathrm{ord}(\beta) = d}} (x - \beta) = \prod_{\substack{d \\ d/n}} Q^{(d)}(x) \tag{6.59}$$

La seconda scomposizione coinvolge i cosiddetti *polinomi ciclotomici* $Q^{(d)}(x)$, le cui radici sono *tutti e soli gli elementi del campo di ordine* d. Il vantaggio derivante dalla loro introduzione è dato dal fatto che per essi è disponibile una formula esplicita basata sulla *funzione aritmetica di Möbius* $\mu(j)$ così definita

$$\mu(j) = \begin{cases} 1 & \text{se} \quad j = 1 \\ (-1)^t & \text{se} \quad j \quad \text{è prodotto di } t \text{ fattori primi distinti} \\ 0 & \text{se} \quad j \quad \text{è prodotto di fattori primi ripetuti} \end{cases} \tag{6.60}$$

La struttura del polinomio ciclotomico è la seguente

$$Q^{(d)}(x) = \prod_{\substack{j \\ j/d}} \left(x^j - 1\right)^{\mu(\frac{d}{j})} = \prod_{\substack{j \\ j/d}} \left(x^{\frac{d}{j}} - 1\right)^{\mu(j)} \tag{6.61}$$

Accanto alla scomposizione di $x^n - 1$ mediante polinomi ciclotomici, esiste però un'altra scomposizione, che raffina la precedente, e che mette in luce i polinomi irriducibili che compongono la $x^{p^m} - x$. Più precisamente vale il seguente teorema (per la dimostrazione si veda [9])

Teorema 6.8.

1. *Il polinomio $x^{p^m} - x$ si scompone nel prodotto di tutti i polinomi monici irriducibili su $GF(p)$ il cui grado d divide m*

$$x^{p^m} - x = \prod_{\substack{d \\ d/m}} m_d(x) \tag{6.62}$$

2. *sia β di ordine d radice di un polinomio $f(x)$ su $GF(p)$; se r è il più piccolo intero tale che $p^r \equiv 1 \pmod{d}$ allora*

$$\beta \quad \beta^p \quad \beta^{p^2} \quad \beta^{p^3} \ldots \beta^{p^{r-1}} \tag{6.63}$$

sono tutte radici distinte di $f(x)$.

Dette radici sono dette *coniugate* di β, mentre r è *l'ordine moltiplicativo* di p modulo d.

È ovvio che l'impiego sinergico delle (6.59), (6.61), (6.63) e (6.62) consente di risolvere il problema connesso con la scomposizione di $x^n - 1$, almeno nel caso in cui sia $n = p^m - 1$ per qualche m e con p primo, mettendo così in luce tutti i polinomi generatori potenzialmente impiegabili per la costruzione di un codice ciclico.

Facciamo subito alcuni esempi.

Esempio 6.17. Cominciamo col caso più semplice, quello con $p^m = 2^2$. In questo esempio e in tutti i successivi con $p = 2$ si può evitare l'uso del del segno meno nella scomposizione dei polinomi e nei vari calcoli. $x^{2^2} + x = x(x^3+1)$ si scompone nel prodotto di tutti i polinomi irriducibili su $GF(2)$ il cui grado divide 2, cioè tutti i polinomi di grado 1 e 2. D'altra parte, per la (6.59), il polinomio x^3+1 si scompone nel prodotto di tutti i polinomi ciclotomici $Q^{(d)}(x)$, con d che divide 3. Usando la (6.61) ricaviamo allora

$$
\begin{aligned}
& x^{2^2} + x = x(x^3+1) = x\, Q^{(1)}(x) Q^{(3)}(x) \\
& Q^{(1)}(x) = x + 1 \\
& Q^{(3)}(x) = (x+1)^{\mu(3)}(x^3+1)^{\mu(1)} = \frac{x^3+1}{x+1} = x^2 + x + 1 \\
& \qquad \text{dunque} \\
& x^{2^2} + x = x(x+1)(x^2+x+1)
\end{aligned}
$$

$Q^{(1)}(x)$ ha come radice l'unico elemento di ordine 1, cioè l'unità. $Q^{(2)}(x)$ contiene invece le radici di ordine 2. Sia allora α primitivo, cioè radice di x^2+x+1, e dunque $\alpha^3 = 1$. Dal teorema 6.7 segue che α^2 ha ordine $3/(3,2) = 3$, e anche α^2 è primitivo. La struttura del campo è riportata in tabella 6.2. ◯

Elementi di $GF(2^2)$				
Come potenza di α	ordine	Radice di	Come 2-pla su $GF(2)$	Come polinomio in α su $GF(2)$
0	$-$	x	00	0
1	1	$x+1$	01	1
α	2	x^2+x+1	10	α
α^2	2	x^2+x+1	11	$1+\alpha$

Tabella 6.2 Struttura del campo $GF(2^2)$.

Esempio 6.18. Consideriamo ora il caso $p^m = 2^3$. Procedendo come prima si ha

$$\begin{aligned}
&x^{2^3} + x = x(x^7 + 1) = xQ^{(1)}(x)Q^{(7)}(x)\\
&Q^{(1)}(x) = x + 1\\
&Q^{(7)}(x) = (x+1)^{\mu(7)}(x^7+1)^{\mu(1)} = \frac{x^7+1}{x+1} =\\
&\qquad = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \quad \text{e dunque}\\
&x^{2^3} + x = x(x+1)(x^6 + x^5 + x^4 + x^3 + x^2 + x + 1)
\end{aligned}$$

Se ora α è un elemento primitivo, si ha $\alpha^7 = 1$, mentre α^k ha ordine $7/(7,k) = 7\ \forall k$, essendo 7 primo. Tutte le altre radici diverse da 0 e 1 sono primitive. Se usiamo l'altra fattorizzazione, il polinomio $x^{2^3} + x$ si scompone nel prodotto di di tutti i polinomi monici irriducibili il cui grado divide 3, cioè di I e III grado. Si ha dunque

$$x^{2^3} + x = (x+1)(x^3 + \ldots)(x^3 + \ldots)$$

Per trovare i due polinomi di terzo grado possiamo notare che $x^3 + x^2 + x + 1$ ha come radice 1, e dunque non è irriducibile. Rimangono $x^3 + x^2 + 1$ e $x^3 + x + 1$ il cui prodotto è proprio $Q^{(7)}(x)$. La scomposizione del campo è allora

$$x^{2^3} + x = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1) \tag{6.64}$$

Si prenda ora α radice di $x^3 + x + 1$, cioè $\alpha^3 = \alpha + 1$. Si ricordi che, a norma del teorema 6.8, se α è radice di un polinomio $f(x)$ allora lo sono anche $\alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3}, \ldots, \alpha^{p^{m-1}}$, che sono le *radici coniugate* di α. Nel nostro caso si ha $m = 3$ e $p = 2$, e dunque $\alpha, \alpha^2, \alpha^{2^2}$ sono tutte le radici di $x^3 + x + 1$; α^3 è invece radice di $x^3 + x^2 + 1$, come si verifica subito

$$\begin{aligned}
(\alpha^3)^3 + (\alpha^3)^2 + 1 &= (\alpha+1)^3 + (\alpha+1)^2 + 1 =\\
&= \alpha^3 + \alpha^2 + \alpha + 1 + \alpha^2 + 1 + 1 = \alpha^3 + \alpha + 1 = 0
\end{aligned}$$

e tali sono anche le sue coniugate $(\alpha^3)^2 = \alpha^6$ e $(\alpha^3)^{2^2} = \alpha^{12} = \alpha^5$. La struttura del campo è riportata in tabella 6.3. ◯

Esempio 6.19. Affrontiamo ora un caso più complesso, quello di $GF(2^4)$. La

Elementi di $GF(2^3)$				
Come potenza di α	ordine	Radice di	Come 3-pla su $GF(2)$	Come polinomio in α su $GF(2)$
0	–	x	000	0
1	1	$x+1$	001	1
α	7	x^3+x+1	010	α
α^2	7	x^3+x+1	100	α^2
α^3	7	x^3+x^2+1	011	$\alpha+1$
α^4	7	x^3+x+1	110	$\alpha^2+\alpha$
α^5	7	x^3+x^2+1	111	$\alpha^2+\alpha+1$
α^6	7	x^3+x^2+1	101	α^2+1

Tabella 6.3 Struttura del campo $GF(2^3)$.

fattorizzazione nei polinomi ciclotomici porta ai seguenti risultati

$$
\begin{aligned}
&x^{2^4}+x = x(x^{15}+1) = xQ^{(1)}(x)Q^{(3)}(x)Q^{(5)}(x)Q^{(15)}(x)\\
&Q^{(3)}(x) = x^2+x+1\\
&Q^{(5)}(x) = (x+1)^{\mu(5)}(x^5+1)^{\mu(1)} = x^4+x^3+x^2+x+1\\
&Q^{(15)}(x) = (x+1)^{\mu(15)}(x^3+1)^{\mu(5)}(x^5+1)^{\mu(3)}(x^{15}+1)^{\mu(1)} =\\
&\qquad = \frac{(x+1)(x^{15}+1)}{(x^3+1)(x^5+1)} = \frac{x^{15}+1}{x^7+x^6+x^5+x^2+x+1} =\\
&\qquad = x^8+x^7+x^5+x^4+x^3+x+1
\end{aligned}
$$

e dunque

$$
\begin{aligned}
x^{2^4}+x = x(x+1)(x^2+x+1)(x^4+x^3+x^2+x+1)\cdot\\
\cdot(x^8+x^7+x^5+x^4+x^3+x+1)
\end{aligned}
$$

Poiché $x^{2^4}+x$ si scompone nel prodotto dei monici irriducibili il cui grado divide 4, cioè di I, II, IV grado, il polinomio $x^8+x^7+x^5+x^4+x^3+x+1$ non può essere irriducibile, ma deriva dal prodotto di due polinomi irriducibili di IV grado. Per trovarli basta considerare che sono comunque nella forma $x^4+ax^3+bx^2+cx+1$, e che non essendo 1 radice del polinomio deve essere $a+b+c=1$; cioè tutti e tre coefficienti sono pari a 1 oppure solo uno dei tre è pari a 1. La prima possibilità fornisce $Q^{(15)}(x)$, ed è scartata. Anche la $x^4+x^2+1=(x^2+x+1)^2$ si scarta e rimangono le altre due, che portano a

$$Q^{(15)}(x) = (x^4+x^3+1)(x^4+x+1)$$

La scomposizione completa è la seguente

$$x^{2^4}+x = x(x+1)(x^2+x+1)(x^4+x^3+x^2+x+1)\cdot \\ \cdot (x^4+x^3+1)(x^4+x+1) \qquad (6.65)$$

Prendiamo ora α primitivo di x^4+x+1; le sue coniugate, radici primitive dello stesso polinomio, sono rispettivamente

$$\alpha, \alpha^2, \alpha^{2^2}=\alpha^4, \alpha^{2^3}=\alpha^8$$

α^3 ha ordine $15/(15,3)=5$ ed è dunque radice di $Q^{(5)}(x)=x^4+x^3+x^2+x+1$; le coniugate sono

$$\alpha^3, (\alpha^3)^2=\alpha^6, (\alpha^3)^{2^2}=\alpha^{12}, (\alpha^3)^{2^3}=\alpha^{24}=\alpha^9$$

α^5 ha ordine $15/(15,5)=3$; esso è radice di $Q^{(3)}(x)=x^2+x+1$ assieme alle sue coniugate che sono

$$\alpha^5, (\alpha^5)^2=\alpha^{10}$$

Infine α^7 ha ordine $15/(15,7)=15$ ed è radice primitiva di x^4+x^3+x+1; le coniugate sono

$$\alpha^7, (\alpha^7)^2=\alpha^{14}, (\alpha^7)^{2^2}=\alpha^{28}=\alpha^{13}, (\alpha^7)^{2^3}=\alpha^{56}=\alpha^{11}$$

La struttura completa di $GF(16)$ è riportata in tabella 6.4. ◯

Esempio 6.20. Mostriamo ora un esempio su un campo di caratteristica 3, analizzando la scomposizione di $x^{3^2}-x$ e mettendo in luce la struttura di $GF(3^2)$. La scomposizione viene fatta sui monici irriducibili su $GF(3)$ di grado 1 e 2 e sui ciclotomici di ordine 1,2,4,8, che sono calcolati nel seguito, ricordando che ora gli elementi di $GF(3)$ sono 0,1 e 2, e che le operazioni vanno fatte $\mod 3$.

$$x^{3^2}-x = x(x^8-1) = xQ^{(1)}(x)Q^{(2)}(x)Q^{(4)}(x)Q^{(8)}(x)$$
$$Q^{(1)}(x) = x-1 = x+2$$
$$Q^{(2)}(x) = (x-1)^{\mu(2)}(x^2-1)^{\mu(1)} = \frac{x^2-1}{x-1} = x+1$$
$$Q^{(4)}(x) = (x-1)^{\mu(4)}(x^2-1)^{\mu(2)}(x^4-1)^{\mu(1)} = \frac{x^4-1}{x^2-1} = x^2+1$$
$$Q^{(8)}(x) = (x-1)^{\mu(8)}(x^2-1)^{\mu(4)}+(x^4-1)^{\mu(2)}(x^8-1)^{\mu(1)} = \frac{x^8-1}{x^4-1} =$$
$$= x^4+1 \quad \text{e la scomposizione porta a}$$
$$x^{3^2}-x = x(x+2)(x+1)(x^2+1)(x^4+1)$$

Elementi di $GF(2^4)$				
Come potenza di α	ordine	Radice di	Come 4-pla su $GF(2)$	Come polinomio in α su $GF(2)$
0	$-$	x	0000	0
1	1	$x+1$	0001	1
α	15	x^4+x+1	0010	α
α^2	15	x^4+x+1	0100	α^2
α^3	5	$x^4+x^3+x^2+x+1$	1000	α^3
α^4	15	x^4+x+1	0011	$\alpha+1$
α^5	3	x^2+x+1	0110	$\alpha^2+\alpha$
α^6	5	$x^4+x^3+x^2+x+1$	1100	$\alpha^3+\alpha^2$
α^7	15	x^4+x^3+x+1	1011	$\alpha^3+\alpha+1$
α^8	15	x^4+x+1	0101	α^2+1
α^9	5	$x^4+x^3+x^2+x+1$	1010	$\alpha^3+\alpha$
α^{10}	3	x^2+x+1	0111	$\alpha^2+\alpha+1$
α^{11}	15	x^4+x^3+x+1	1110	$\alpha^3+\alpha^2+\alpha$
α^{12}	5	$x^4+x^3+x^2+x+1$	1111	$\alpha^3+\alpha^2+\alpha+1$
α^{13}	15	x^4+x^3+x+1	1101	$\alpha^3+\alpha^2+1$
α^{14}	15	x^4+x^3+x+1	1001	α^3+1

Tabella 6.4 Struttura del campo $GF(2^4)$.

Tuttavia $Q^{(8)}(x)$ deve essere riducibile, poiché il grado massimo dei polinomi irriducibili è 2. Per trovare i due polinomi irriducibili di secondo grado che compongono $Q^{(8)}(x)$ consideriamo la generica forma di tale polinomio x^2+ax+b e proviamo tutte le possibilità, riportate nella tabella sotto, tenendo conto che $x^2+1=Q^{(4)}(x)$ è l'altro irriducibile di II grado, e che per verificare l'irriducibilità basta controllare se $x=0, x=1$, o $x=2$ sono radici, cioè se il polinomio è divisibile per $x, x+2$ o $x+1$.

a	b	x^2+ax+b	$radici$
0	0	x^2	$x=0$
0	1	x^2+1	$irr. \Rightarrow Q^{(4)}(x)$
0	2	x^2+2	$x=1, x=2$
1	0	x^2+x	$x=0, x=2$
1	1	x^2+x+1	$x=1$
1	2	x^2+x+2	$irr.$
2	0	x^2+2x	$x=0, x=1$
2	1	x^2+2x+1	$x=2$
2	2	x^2+2x+2	$irr.$

La scomposizione completa è allora la seguente

$$x^{3^2} - x = x(x+2)(x+1)(x^2+1)(x^2+x+2)(x^2+2x+2) \qquad (6.66)$$

Prendiamo α primitivo di ordine 8, radice assieme ad α^3 di x^2+x+2. α^2 ha ordine $8/(8,2) = 4$ ed è radice di $Q^{(4)}(x) = x^2+1$ assieme alla coniugata $(\alpha^2)^3 = \alpha^6$. α^5 ha ordine $8/(8,5) = 8$ e assieme a α^7 è radice di x^2+2x+2 che compone $Q^{(8)}(x)$. Elevando a potenza α e tenendo conto che $\alpha^2+\alpha+2=0$, cioè $\alpha^2 = 2\alpha+1$, si ottengono le rappresentazioni di tutti gli elementi come polinomi in α di grado ≤ 1, secondo quanto esposto nella tabella 6.5. ◯

Elementi di $GF(3^2)$				
Come potenza di α	ordine	Radice di	Come 2-pla su $GF(3)$	Come polinomio in α su $GF(3)$
0	–	x	00	0
1	1	$x+2$	01	1
α	8	x^2+x+2	10	α
α^2	4	x^2+1	21	$2\alpha+1$
α^3	8	x^2+x+2	22	$2\alpha+2$
α^4	2	$x+1$	02	2
α^5	8	x^2+2x+2	20	2α
α^6	4	x^2+1	12	$\alpha+2$
α^7	8	x^2+2x+2	11	$\alpha+1$

Tabella 6.5 Struttura del campo $GF(3^2)$.

6.4.2 Alcuni esempi di codici ciclici

Torniamo ora alla costruzione di un codice ciclico basata sulla scomposizione di $x^n - 1$ e facciamo alcuni esempi concreti basati sulle scomposizioni viste precedentemente.

Esempio 6.21. Consideriamo il caso $n = 2^3 - 1 = 7$. Dalla scomposizione (6.64) scegliamo $g(x) = x^3 + x + 1$. Si ha $n-k = 3$ e $k = 4$. Si tratta di un codice $C(7,4)$. La matrice di controllo è

$$\boldsymbol{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Dalla $\boldsymbol{G}$ si ricava che le parole di codice sono, oltre alla nulla e alla unitaria, 1101000 e 1110010 con le rispettive 6 traslate. La scomposizione (6.64) consente di ricavare il polinomio di controllo e la struttura della matrice $\boldsymbol{H}$; si ha infatti

$$\begin{aligned} x^7+1 = g(x)h(x) &= (x^3+x+1)(x+1)(x^3+x^2+1) = \\ &= (x^3+x+1)(x^4+x^2+x+1) \end{aligned}$$

che porta alla seguente matrice di controllo

$$\boldsymbol{H} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Si riconosce subito la struttura della matrice di controllo di un codice di Hamming $C(7,4)$, poiché come colonne ci sono tutte le possibili terne (tranne la nulla), ciascuna presa una sola volta. Il codice di Hamming è dunque un codice ciclico generato da $g(x) = x^3 + x + 1$. Tuttavia, anche scegliendo come generatore l'altro polinomio iriducibile di terzo grado si perviene nuovamente a un codice di Hamming, con le colonne della matrice $\boldsymbol{H}$ permutate.

La scelta $g(x) = (x^3+x+1)(x^3+x^2+1) = x^6+x^5+x^4+x^3+x^2+x+1$ porta invece alla matrice $\boldsymbol{G} = (1111111)$ di un codice a ripetizione $C(7,1)$. Il suo codice duale si ricava da $g(x) = x+1$. In questo caso si ha

$$\boldsymbol{G} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

con $\boldsymbol{H} = (1111111)$. L'equazione di controllo viene soddisfatta solo da vettori di peso pari; essi sono in numero di $\binom{7}{6} + \binom{7}{4} + \binom{7}{2} + \binom{7}{0} = 64$. ◯

Esempio 6.22. Consideriamo ora il caso $n = 2^4 - 1 = 15$. Dalla scomposizione (6.65), scegliendo $g(x) = (x^4+x+1)(x^4+x^3+x^2+x+1) = x^8+x^7+x^6+x^4+1$ si ha $n-k = 8$ e $k = 7$. Si tratta di un codice $C(15,7)$. La matrice generatrice è

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & & & & & & \\ & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & & & & & \\ & & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & & & & \\ & & & \cdot & & & & & & & & \cdot & & & \\ & & & & \cdot & & & & & & & & \cdot & & \\ & & & & & \cdot & & & & & & & & \cdot & \\ & & & & & & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Per la matrice di controllo si ricava $h(x) = (x^4 + x^3 + 1)(x^2 + x + 1)(x + 1) = x^7 + x^6 + x^4 + 1$

$$\boldsymbol{H} = \begin{pmatrix} & & & & & & & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ & & & & & & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & \\ & & & & & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & & \\ & & & & \cdot & & & & & & & \cdot & & & \\ & & & \cdot & & & & & & & \cdot & & & & \\ & & \cdot & & & & & & & \cdot & & & & & \\ & \cdot & & & & & & & \cdot & & & & & & \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & & & & & & & \end{pmatrix}$$

Il codice in questione è un codice BCH 2-correttore, anche se la matrice di controllo non è riconoscibile in modo immediato. Nel prossimo paragrafo metteremo in luce un approccio che ci consentirà di giustificare tale affermazione.◯

Esempio 6.23. Come esempio finale consideriamo la scomposizione di $x^{3^2} - x$ riportata nella (6.66). Scegliamo $g(x) = (x^2 + x + 2)(x^2 + 1) = x^4 + x^3 + x + 2$. La matrice generatrice che si ricava è

$$\boldsymbol{G} = \begin{pmatrix} 2 & 1 & 0 & 1 & 1 & & & \\ & 2 & 1 & 0 & 1 & 1 & & \\ & & 2 & 1 & 0 & 1 & 1 & \\ & & & 2 & 1 & 0 & 1 & 1 \end{pmatrix}$$

e fornisce in tutto $3^4 = 81$ parole di codice. Con l'aiuto della matrice di controllo

$$\boldsymbol{H} = \begin{pmatrix} & & & 1 & 2 & 1 & 1 & 1 \\ & & 1 & 2 & 1 & 1 & 1 & \\ & 1 & 2 & 1 & 1 & 1 & & \\ 1 & 2 & 1 & 1 & 1 & & & \end{pmatrix}$$

ricavata dal polinomio $h(x) = (x^2 + 2x + 2)(x + 2)(x + 1) = x^4 + 2x^3 + x^2 + x + 1$, si può verificare che le parole di codice sono costituite da 9 gruppi di 8 parole, ottenute mediante permutazione ciclica di una qualunque parola del gruppo, da 3 gruppi di 2 parole (precisamente le 10101010, 20202020, 12121212 con le corrispondenti traslazioni) e dalle 00000000, 11111111, e 22222222. Il peso minimo delle parole di codice è 4 e il codice corregge 1 errore. ◯

6.4.3 Radici caratterizzanti di un codice ciclico

Daremo ora un'altra caratterizzazione dei codici ciclici, basata sull'assegnazione delle radici delle parole di codice, le quali individuano indirettamente il polinomio generatore e quindi il codice. Questo approccio consente di mettere in luce alcune proprietà strutturali della matrice di controllo che faciliteranno in

modo notevole le operazioni automatiche di rilevamento e correzione degli errori. Nella sostanza progetteremo il codice imponendo che un certo elemento α sia radice del polinomio $g(x)$ e quindi della parola di codice; dunque

$$a(x) \in \mathfrak{C} \quad \text{se e solo se} \quad a(\alpha) = 0 \quad \text{cioè} \quad \boldsymbol{H}\boldsymbol{a}^T = 0$$

con $\boldsymbol{H}$ matrice di controllo. Prendendo α, a norma del teorema 6.8, acquisiamo implicitamente anche tutte le sue coniugate $\alpha^p, \alpha^{p^2}, \alpha^{p^3}, \ldots, \alpha^{p^{r-1}}$; se inoltre imponiamo che $\alpha_1, \alpha_2, \ldots, \alpha_t$ siano radici, il polinomio generatore deve contenere tutti i polinomi $m_1(x), m_2(x), \ldots, m_t(x)$, con α_i radice di $m_i(x)$. Di conseguenza

$$g(x) = \text{m.c.m.}\,(m_1(x), m_2(x), \ldots, m_t(x))$$

Se le $\alpha_j, 1 \leq j \leq t$ sono radici di $a(x)$, si deve avere

$$a(\alpha_j) = a_0 + a_1\alpha_j + \ldots + a_{n-1}\alpha_j^{n-1} = 0 \qquad 1 \leq j \leq t$$

che può essere scritta mettendo in evidenza la struttura della matrice di controllo

$$\boldsymbol{H} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{n-1} \\ & \vdots & & & \\ 1 & \alpha_t & \alpha_t^2 & \ldots & \alpha_t^{n-1} \end{pmatrix} \tag{6.67}$$

dove gli elementi α_j possono essere scritti usando la loro rappresentazione in $GF(p^m)$ come m-ple p-arie associate ai coefficienti dei polinomi di grado $< m$ in α, radice primitiva.

Esempio 6.24. Come esempio si cerchi il codice per cui α è radice di x^3+x+1 su $GF(2)$. Il polinomio ha tre radici, $\alpha, \alpha^2, \alpha^{2^2}$. Per la seconda parte del teorema 6.8 si ha $r = 3$ e $2^3 \equiv 1 \pmod 7$, e quindi $n = 7$. Siamo in $GF(2^3)$ con α primitivo. La matrice di controllo è

$$\boldsymbol{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

che corrisponde alla matrice (6.36) di un codice di Hamming $C(7,4)$, con un diverso ordinamento delle colonne.

Se ora imponiamo che anche α^3 sia radice si ottiene invece

$$\boldsymbol{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

che è il codice BCH $C(7,1)$ descritto dalle matrici (6.44) e (6.45).

Prendiamo ora α primitivo di $GF(2^4)$, e quindi $\alpha^{15}=1$. Se imponiamo che $g(x)$ abbia come radici α e α^3 i coniugati sono

$$\begin{array}{llll} \alpha & \alpha^2 & \alpha^4 & \alpha^8 \\ \alpha^3 & \alpha^6 & \alpha^{12} & \alpha^{24}=\alpha^9 \end{array} \qquad \begin{array}{ll} m_1(x) & \text{grado } 4 \\ m_3(x) & \text{grado } 4 \end{array}$$

dove con $m_i(x)$ si indicano i polinomi di cui α^i è radice. Essi si possono ricavare dalla tabella 6.4 e sono rispettivamente $m_1(x)=x^4+x+1, m_3(x)=x^4+x^3+x^2+x+1$. Il polinomio generatore è pari al prodotto dei due polinomi: $g(x)=(x^4+x+1)(x^4+x^3+x^2+x+1)$. La matrice (6.67) prende allora la seguente forma

$$\boldsymbol{H}=\begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} \end{pmatrix}$$

che corrisponde alla matrice di controllo di un codice BCH $C(15,7)$.

Imponendo come radice anche α^5 si fa intervenire $m_5(x)=x^2+x+1$. Il polinomio generatore è pari al prodotto $g(x)=m_1(x)m_3(x)m_5(x)=(x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)$ mentre la matrice di controllo diventa

$$\boldsymbol{H}=\begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \dots & \alpha^6 & \alpha^9 & \alpha^{12} \\ 1 & \alpha^5 & \alpha^{10} & 1 & \dots & 1 & \alpha^5 & \alpha^{10} \end{pmatrix}$$

○

Si noti che, per effetto del teorema 6.8, gli elementi di un campo $GF(p^m)$ si distibuiscono in sottinsiemi disgiunti che sono costituiti da tutte le radici coniugate relative a un certo polinomio detto *polinomio minimo*. Se α^s è radice di $m_s(x)$, lo sono anche $(\alpha^s)^p, (\alpha^s)^{p^2}, (\alpha^s)^{p^3}, \dots$. Facendo riferimento per semplicità ai soli esponenti costruiamo allora l'insieme degli interi

$$s \quad sp \quad sp^2 \quad sp^3 \quad \dots \quad sp^{r-1} \tag{6.68}$$

dove r è il più piccolo intero tale che $sp^r \equiv s \mod (p^m-1)$. L'insieme degli interi generato in questo modo dall'esponente s è il cosiddetto *laterale ciclotomico* relativo alla radice α^s, e la costruzione sistematica di tutti i laterali ciclotomici associati a tutti gli interi $0 \le s < p^m-1$ fornisce immediatamente informazioni sulla struttura del campo $GF(p^m)$, cioè sull'ordine delle sue radici e sulla struttura dei polinomi a queste associati. Come esempio riconsideriamo il caso di $GF(2^4)$ partendo da un elemento α primitivo ($\alpha^{15}=1$). Il laterale ciclotomico L_1 relativo ad α contiene i quattro interi $1,\ 2,\ 2^2=4,\ 2^3=8$. Poiché l'ordine moltiplicativo di 2 modulo 15 è 4, si ha $2^4 \equiv 1$, e il laterale termina appunto con $2^3=8$. Prendiamo ora il più piccolo esponente non

compreso nella lista, che è 3. Il laterale ciclotomico L_3 è costituito dagli interi $3,\ 3\cdot 2=6,\ 3\cdot 2^2=12,\ 3\cdot 2^3=24\equiv 9(\mod 15)$. Proseguendo come sopra e facendo le riduzioni $\mod(15)$ si ottengono complessivamente i seguenti laterali ciclotomici

$$\begin{array}{lll} L_0=\{0\} & m_0(x) & \text{grado } 1\\ L_1=\{1,2,4,8\} & m_1(x) & \text{grado } 4\\ L_3=\{3,6,12,9\} & m_3(x) & \text{grado } 4\\ L_5=\{5,10\} & m_5(x) & \text{grado } 2\\ L_7=\{7,14,13,11\} & m_7(x) & \text{grado } 4 \end{array}$$

Nel caso ternario $GF(3^2)$ di tabella 6.5 ci sono i seguenti 5 laterali ciclotomici

$$\begin{array}{lll} L_0=\{0\} & m_0(x) & \text{grado } 1\\ L_1=\{1,3\} & m_1(x) & \text{grado } 2\\ L_2=\{2,6\} & m_2(x) & \text{grado } 2\\ L_4=\{4\} & m_4(x) & \text{grado } 1\\ L_5=\{5,7\} & m_5(x) & \text{grado } 2 \end{array}$$

Un metodo generale per assegnare un codice ciclico attraverso le sue radici è allora il seguente

Assegnazione di un codice ciclico mediante le sue radici. Si scelga una radice n-esima dell'unità, β, imponendo che il polinomio generatore $g(x)$ sia costituito dai polinomi minimi di $\beta,\beta^2,\ldots,\beta^{d-1}$

$$\begin{array}{l} \beta\in GF(p^m)\quad \text{con}\quad p^m\equiv 1 \mod n\\ g(x)\ \text{ prodotto dei polinomi minimi di } \beta,\beta^2,\ldots,\beta^{d-1} \end{array}$$

Nel caso in cui β sia primitivo il codice si definisce primitivo. d è la *distanza di progetto* .

Vale il seguente

Teorema 6.9. *La distanza minima di un codice ciclico assegnato mediante le sue radici n-esime dell'unità $\beta,\beta^2,\ldots,\beta^{d-1}$ è maggiore o uguale alla distanza di progetto d.*

Dim. Bisogna verificare che per qualunque scelta di $d_{min}-1$ colonne, queste sono sempre linearmente indipendenti. Prendiamo la matrice (6.67) del codice e scriviamola sulla base della radice β

$$\boldsymbol{H}=\begin{pmatrix} 1 & \beta & \beta^2 & \ldots & \beta^{n-1}\\ 1 & \beta^2 & \beta^4 & \ldots & \beta^{(n-1)2}\\ & \vdots & & & \\ 1 & \beta^{d-1} & \beta^{2(d-1)} & \ldots & \beta^{(n-1)(d-1)} \end{pmatrix} \tag{6.69}$$

Prendiamo ora $d_{min}-1$ colonne qualunque e verifichiamo che sono linearmente indipendenti, ponendo $\xi_j = \beta^{i_j}, 1 \leq j \leq d-1$, al fine di semplificare la notazione. Si ricava la sottomatrice

$$\begin{pmatrix} \xi_1 & \xi_2 & \dots & \xi_{d-1} \\ \xi_1^2 & \xi_2^2 & \dots & \xi_{d-1}^2 \\ \vdots & \vdots & & \vdots \\ \xi_1^{d-1} & \xi_2^{d-1} & \dots & \xi_{d-1}^{d-1} \end{pmatrix}$$

Se ora il determinante è diverso da zero si ha la tesi. Ma

$$\det \begin{pmatrix} \xi_1 & \xi_2 & \dots & \xi_{d-1} \\ \xi_1^2 & \xi_2^2 & \dots & \xi_{d-1}^2 \\ \vdots & \vdots & & \vdots \\ \xi_1^{d-1} & \xi_2^{d-1} & \dots & \xi_{d-1}^{d-1} \end{pmatrix} = \xi_1\xi_2\dots\xi_{d-1} \det \begin{pmatrix} 1 & 1 & 1 & 1 \\ \xi_1 & \xi_2 & \dots & \xi_{d-1} \\ \vdots & \vdots & & \vdots \\ \xi_1^{d-2} & \xi_2^{d-2} & \dots & \xi_{d-1}^{d-2} \end{pmatrix}$$

$$= \xi_1\xi_2\dots\xi_{d-1} \prod_{i>j} (\xi_i - \xi_j) \neq 0$$

poiché il determinante è di Vandermonde; inoltre gli ξ_i non sono mai nulli e $\xi_i \neq \xi_j$ se $i \neq j$, in quanto β è una radice n-esima dell'unità, e dunque $\beta^0, \beta^1, \beta^2, \beta^{n-1}$ sono tutte radici distinte □

L'assegnazione delle radici può essere semplificata tenendo conto che in generale

$$m_i(x) = m_{pi}(x)$$

Nel caso binario, per correggere fino a t errori, basta allora assegnare le radici con esponente dispari

$$\beta \quad \beta^3 \quad \beta^5 \quad \beta^{2t-1}$$

poiché il teorema sulla distanza di progetto ci assicura che $d_{min} \geq 2t+1$. I codici che se ne ricavano sono detti BCH t-correttori e offrono prestazioni interessanti, soprattutto per valori dei parametri non troppo elevati. Tuttavia le prestazioni asintotiche non sono incoraggianti, in quanto se la distanza di progetto rimane costante si ha

$$R_{tBCH} = \frac{k}{n} = \frac{n-mt}{n} \longrightarrow 1 \qquad \lambda_{tBCH} = \frac{d_{min}}{n} \longrightarrow 0 \tag{6.70}$$

Tuttavia si potrebbe dimostrare che in generale non esistono successioni asintotiche di codici t-BCH con un tasso e una capacità di correzione strettamente maggiori di zero (si veda [59]).

Esempio 6.25. Si voglia costruire un codice BCH che corregge 4 errori con $n = 31$. La distanza minima dev'essere $d_{min} \geq 9$ e quindi la distanza di progetto è pari a 8. Il codice deve contenere $\beta, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6, \beta^7, \beta^8$ come radici. In realtà, per quanto visto precedentemente bastano le radici dispari, cioè $\beta, \beta^3, \beta^5, \beta^7$, e il codice è generato dal polinomio $g(x) = m_1(x) \cdot m_3(x) \cdot m_5(x) \cdot m_7(x)$. I laterali ciclotomici associati sono i seguenti

$$\begin{array}{lll} L_1 = \{1, 2, 4, 8, 16\} & m_1(x) & \text{grado } 5 \\ L_3 = \{3, 6, 12, 24, 17\} & m_3(x) & \text{grado } 5 \\ L_5 = \{5, 10, 20, 9, 18\} & m_5(x) & \text{grado } 5 \\ L_7 = \{7, 14, 28, 25, 19\} & m_7(x) & \text{grado } 5 \end{array}$$

Si noti che in questo caso l'introduzione di $m_5(x)$ ha forzato anche la presenza di β^9; ciò comporta una distanza minima pari a 11. La distanza minima è allora strettamente maggiore di quella di progetto, il codice corregge 5 errori e il grado di $g(x)$ è pari a 20. ◯

Come considerazione finale facciamo notare che anche i codici di Golay, studiati nel paragrafo 6.3.6, si possono introdurre come codici ciclici. Nel caso del codice binario $C(23, 2^{12}, 7)$ si può notare che $2^{11} - 1 = 89 \cdot 23$, e dunque $2^{11} \equiv 1 \mod 23$. Possiamo allora individuare $\beta \in GF(2^{11})$, con $\beta = \alpha^{89}$, come radice 23-esima dell'unità. I laterali caratterizzanti sono

$$\begin{array}{lll} L_1 = \{1, 2, 4, 8, 16, 9, 18, 13, 3, 6, 12\} & m_1(x) & \text{grado } 11 \\ L_5 = \{5, 10, 20, 17, 11, 22, 21, 19, 15, 7, 14\} & m_5(x) & \text{grado } 11 \end{array}$$

Si può verificare che la scomposizione di $x^{23} + 1$ è data da

$$x^{23} + 1 = Q^{(1)}(x)Q^{(23)}(x) = (x+1)(x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1) \cdot \\ \cdot (x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1)$$

Il codice di Golay può essere progettato come codice ciclico fissando una capacità di correzione di 2 errori, cioè $d_{min} \geq 5$. Ciò impone la presenza di $\beta, \beta^2, \beta^3, \beta^4$, cioè β, β^3 essendo il codice binario. Essi stanno entrambi in L_1, e dunque $g(x) = m_1(x)$ di grado 11 è generatore di un codice equivalente al Golay $C(23, 2^{12}, 7)$. Anche in questo caso la vera distanza minima è maggiore di quella di progetto, poiché sappiamo che in realtà si ha $d_{min} = 7$.

Per il codice ternario Golay $C(11, 3^6, 5)$ si può fare un discorso analogo, tenuto conto che $3^5 \equiv 1 \mod 11$. I due laterali ciclotomici sono

$$\begin{array}{lll} L_1 = \{1, 3, 9, 5, 4\} & m_1(x) & \text{grado } 5 \\ L_2 = \{2, 6, 7, 10, 8\} & m_2(x) & \text{grado } 5 \end{array}$$

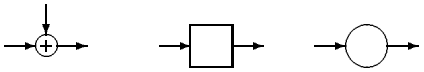
e la scomposizione di $x^{11} - 1$ che si ricava è la seguente

$$x^{11} - 1 = Q^{(1)}(x)Q^{(11)}(x) = (x+2)(x^5 + x^4 + 2x^3 + x^2 + 2) \cdot \\ \cdot (x^5 + 2x^3 + x^2 + 2x + 2)$$

6.5 Circuiti per la codifica/decodifica cablata

Dal punto di vista applicativo i codici ciclici appena analizzati hanno il grosso pregio di consentire operazioni efficienti di codifica/decodifica, basate essenzialmente sull'impiego di *reti logiche sequenziali* (RLS) che attuano operazioni *cablate* di somma e prodotto tra elementi di un campo finito $GF(q) = GF(p^k)$. Per operazione cablata s'intende ch'essa viene relizzata direttamente con le unita logiche minime disponibili a livello circuitale, cioè gli elementi AND, OR, NOT o con le funzioni universali NAND o NOR.

In generale una rete logica sequenziale che effettua operazioni su $GF(q)$ è costituita da tre elementi base: una *cella di memoria* q-aria che è in grado di contenere un qualunque elemento del campo, un *sommatore* che accetta in ingresso due elementi del campo e fornisce in uscita la somma dei due, e un *moltiplicatore* per una certa costante.



Poiché ogni elemento di $GF(p^k)$ si può esprimere in modo opportuno facendo riferimento ai soli interi del campo (si ricordino p.es. le rappresentazioni contenute nelle tabelle 6.5 e 6.4), in pratica ci si può riferire direttamente a celle di memoria, sommatori e moltiplicatori che lavorano su $GF(p)$. Ricordiamo inoltre che il prodotto tra elementi di $GF(p^k)$ può essere pensato anche come prodotto tra polinomi modulo un polinomio irriducibile. Possiamo dunque ricorrere a una o all'altra delle interpretazioni sulla base della nostra convenienza.

Il primo esempio che proponiamo è relativo a un circuito atto alla moltiplicazione tra un polinomio $a(x) = a_k x^k + a_{k-1} x^{k-1} + \ldots + a_0$ e un polinomio fisso $m(x) = m_r x^r + m_{r-1} x^{r-1} + \ldots + m_0$. Il circuito riportato in fig 6.2 si com-

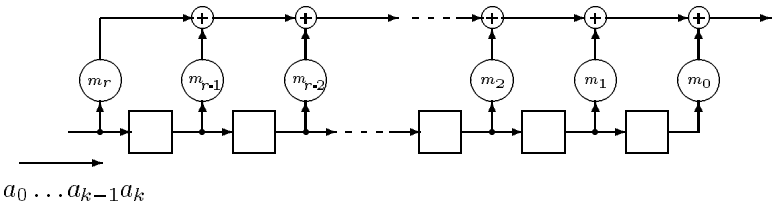


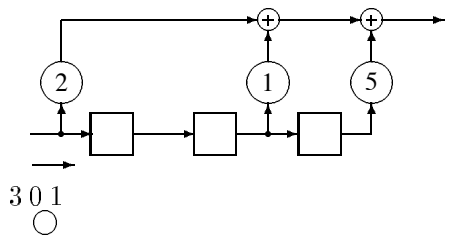
Figura 6.2 Circuito per la moltiplicazione di un polinomio $a(x)$ per il polinomio $m(x)$.

pone di r celle di memoria, il cui contenuto viene traslato nella cella successiva (seguendo il verso delle frecce) ad istanti di tempo prefissati e controllati da un orologio interno. La composizione della figura è anche nota col nome di *registro*. Supponiamo che l'ingresso sia alimentato con i coefficienti del polinomio $a(x)$ a partire da quello relativo al grado massimo e che il registro venga inizializzato

inserendo degli zeri nelle celle di memoria. Quando entra il primo termine, a_k, questi viene moltiplicato per il coefficiente m_r e il risultato di tale prodotto va direttamente in uscita. Poiché il contenuto di tutte le celle è ancora zero, $a_k m_r$ è l'uscita dopo il primo istante. Essa coincide col coefficiente di grado massimo $k+r$ nel prodotto $a(x) \cdot m(x)$, ed è quindi il primo coefficiente in uscita.

Nel frattempo, però, a_k è stato memorizzato nella prima cella. Al passo successivo compare in ingresso a_{k-1}, che prima di uscire viene moltiplicato per m_r. In uscita questo contributo si somma a $a_k m_{r-1}$, derivante dalla prima cella, producendo a $a_k m_{r-1} + a_{k-1} m_r$, che nel prodotto $a(x) \cdot m(x)$ è il coefficiente relativo a x^{k+r-1}. Il procedimento continua fino a quando le r celle di memoria hanno assorbito ed espulso tutti i coefficienti di $a(x)$. In particolare, dopo $k+r$ passi della moltiplicazione tutte le celle sono nuovamente vuote (si suppone che i coefficienti di $a(x)$ siano seguiti da degli zeri), tranne l'ultima che contiene a_0; questo coefficiente va in uscita moltiplicato per r_0, determinando così il termine noto $a_0 r_0$ nel prodotto tra $a(x)$ e $m(x)$.

Esempio 6.26. Moltiplichiamo $a(x) = x^2 + 3$ per $m(x) = 2x^3 + x + 5$. Il prodotto vale $2x^5 + 7x^3 + 5x^2 + 3x + 15$



Ingresso			Registro			Uscita
3	0	1	0	0	0	2
	3	0	1	0	0	0
		3	0	1	0	7
			3	0	1	5
			0	3	0	3
			0	0	3	15

La struttura del registro moltiplicatore può essere sfruttata per costruire un *registro lineare a scorrimento retroazionato (RLSR)*, che chiameremo di *tipo 1*, descritto nella figura 6.3. In questo circuito l'uscita viene riportata all'ingresso mediante una *retroazione*; se l'ingresso esterno è nullo l'evoluzione del registro diventa allora autonoma e dipendente solo dallo stato iniziale e dalla rete di retrozione. Poichè il numero di stati è finito e la macchina è deterministica, si

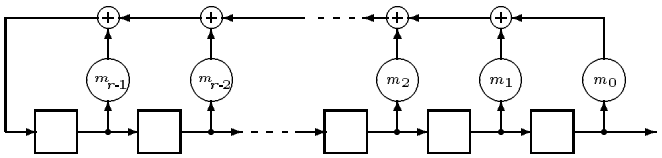


Figura 6.3 Registro lineare a scorrimento retroazionato di tipo 1.

ottiene una *sequenza periodica* degli stati, e quindi una sequenza periodica in

uscita. La periodicità delle sequenze, il legame tra periodo e stato iniziale e il numero possibile di periodi saranno studiati nella sezione 8.2.2, nella quale i *RLSR* saranno considerati dal punto di vista della pseudocasualità delle sequenze generate.

Un'altra possibilità per costruire un circuito atto a moltilicare due polinomi è riportato in figura 6.4. In questo caso il primo coefficiente in ingresso,

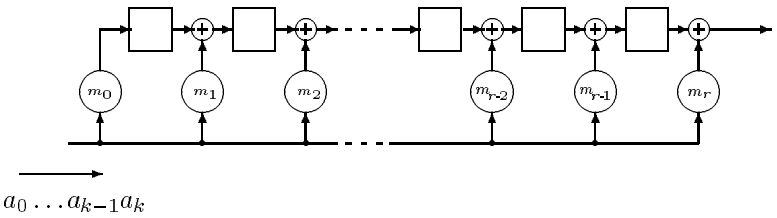


Figura 6.4 Altro circuito per la moltiplicazione tra due polinomi.

a_k, viene moltiplicato per $m_0, m_1, \ldots, m_{r-1}$, per poi finire all'interno delle r celle di memoria, mentre il prodotto $a_k m_r$ finisce direttamente in uscita come coefficiente di grado massimo relativo al prodotto $a(x)m(x)$. Al passo successivo finisce in uscita $a_k m_{r-1}$, che è il contenuto dell'ultima cella al passo precedente, sommato con $a_{k-1} m_r$, che deriva dall'ingresso di a_{k-1} nel registro, cioè $a_k m_{r-1} + a_{k-1} m_r$. In pratica ad ogni passo a_i viene moltiplicato per $m(x)$. All'ultimo passo il registro è vuoto in tutte le sue celle meno che nell'ultima, che contiene $a_0 m_0$, termine noto del prodotto $a(x)m(x)$.

Invertendo il funzionamento del moltiplicatore, e quindi usando l'ingresso come uscita e viceversa, o se si preferisce retrozionando l'uscita sull'ingresso, si ottiene un circuito adatto alla divisione di un polinomio $a(x)$ per $d(x)$ prefissato. Nella divisione tra $a(x) = a_k x^k + \ldots$ e $d(x) = d_r x^r + \ldots$ il primo coefficiente

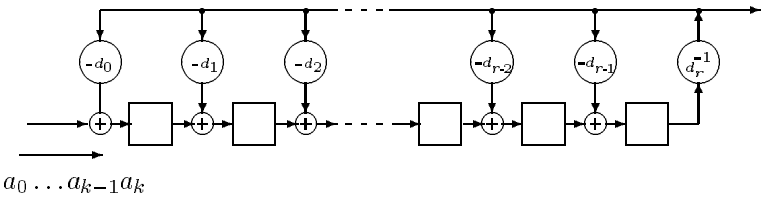
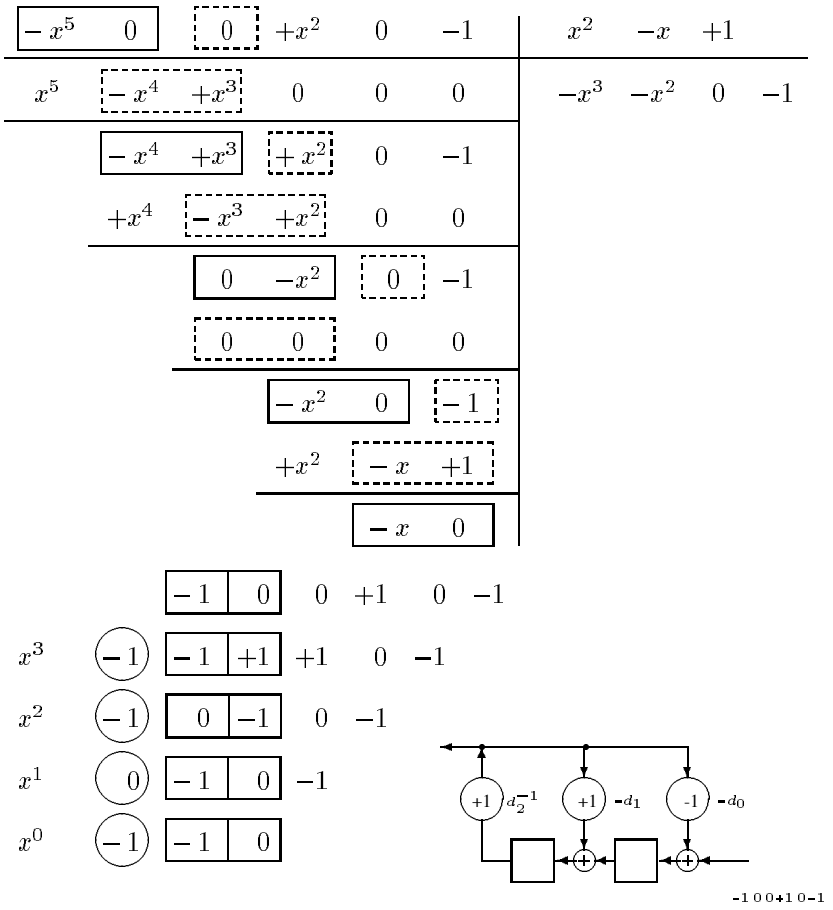


Figura 6.5 Circuito per la divisione del polinomio $a(x)$ per $d(x)$.

del quoziente, relativo al termine x^{k-r}, vale $a_k d_r^{-1}$; successivamente bisogna sottrarre da $a(x)$ il polinomio $a_k d_r^{-1} d(x)$. Questo è esattamente quanto compie il circuito di figura 6.5, quando dopo r passi il termine a_k esce dall'ultima cella di memoria. Successivamente a ogni passo viene sottratto il termine $q_j d(x)$, con q_j il coefficiente del quoziente che va in uscita.

Esempio 6.27. Consideriamo $a(x) = -x^5 + x^2 - 1$ e $d(x) = x^2 - x + 1$ con coeffcienti $-1, 0, +1$ in $GF(3)$. Nella divisione rappresentata secondo le consuete regole i rettangoli con linea continua evidenziano i successivi stati del registro per la divisione, mentre quelli tratteggiati il contenuto della rete di retroazione. I rettangoli tratteggiati più piccoli rappresentano invece il prossimo ingresso del registro.



○

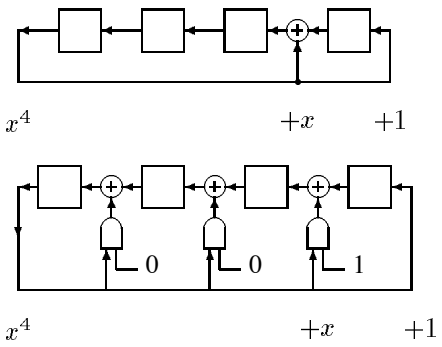
Poiché lo stato successivo del registro dipende deterministicamente da quello precedente secondo una rete di retroazione, possiamo far evolvere il registro in modo autonomo, facendolo funzionare come *registro lineare a scorrimento retroazionato* di *tipo 2*; in tal caso bisogna partire da un qualsiasi stato iniziale diverso dallo stato nullo scollegando l'ingresso. Se $c^{(t)}(x) = c_0 + c_1 x + \ldots +$

$c_{r-1}x^{r-1}$ è il polinomio associato all'istante t al contenuto del registro, al passo successivo, dopo una traslazione verso destra, il contenuto diventa

$$c^{(t+1)}(x) = c_0 x + c_1 x^2 + \ldots + c_{r-2}x^{r-1} - c_{r-1}\left(d_r^{-1}d(x) - x^r\right) = \\ = xc^{(t)}(x) - c_{r-1}d_r^{-1}d(x)$$

da cui si evince che il nuovo polinomio $c^{(t+1)}(x)$ è della stessa classe di $xc^{(t)}(x)$ modulo $d(x)$. Una traslazione verso destra, e quindi un passo dell'evoluzione autonoma, corrispondono allora a una moltiplicazione del polinomio iniziale per x modulo $d(x)$. Questo fatto può essere sfruttato convenientemente per fare operazioni su un campo finito $GF(p^k)$; in particolare si può metter in evidenza una rappresentazione del campo associata a una certa radice α primitiva e radice di $d(x)$.

Prendiamo come esempio il caso di $GF(2^4)$, con α radice di $d(x) = x^4 + x + 1$, cioè $\alpha^4 = \alpha + 1$. Il registro associato al polinomio $d(x)$ è il seguente Si noti che nel caso binario il valore (1 o 0) dei coefficienti d_i relativi a x^i



determina la presenza o l'assenza di una connessione nella rete di retrozione in corrispondenza alla posizione di x^i. Per poter usare lo stesso circuito di base qualunque sia il polinomio associato alla rete di retrozione, si può usare una porta AND come riportato nella stessa figura.

Facendo evolvere il registro a partire dallo stato iniziale 0001, che corrisponde alla rappresentazione dell'unità, il contenuto del registro viene moltiplicato successivamente per α, eseguendo nel contempo la riduzione modulo $\alpha^4+\alpha+1$. La successione degli stati del registro è quella riportata nella tabella dell'esempio 6.19, che corrisponde anche alla successione $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{14}$; quando si arriva a $\alpha^{15} = 1$ il ciclo ricomincia.

La modalità di retroazione, che sottrae a ogni passo il polinomio $c_{r-1}d_r^{-1}d(x)$ con c_{r-1} contenuto del'ultima cella, è in questo caso diversa da quella vista nel registro di tipo 1 in figura 6.3, dove invece la retroazione costituita dalla somma

del contenuto di tutte le celle (secondo la struttura del polinomio $m(x)$) viene riportata sulla cella d'ingresso. Dal punto di vista applicativo queste due tipologie di registri vengono impiegate prevalentemente nella generazione di sequenze periodiche (soprattutto pseudocasuali) e nelle operazioni di codifica/decodifica per codici ciclici, come vedremo fra poco.

6.5.1 Circuiti di codifica per un codice ciclico

Useremo ora le proprietà dei registri a scorrimento retroazionato per costruire dei circuiti che realizzano la codifica (e decodifica) di tipo cablato; in altre parole le computazioni relative alla determinazione delle cifre di controllo (codifica) e della individuazione e correzione dell'errore (decodifica) verranno effettuate direttamente a livello circuitale elementare, impiegando cioè i sommatori, le celle di memoria e i moltiplicatori visti precedentemente. Lavorando in binario si ha un'ulteriore semplificazione, giacché il moltiplicatore può valere solo 0, che corrisponde all'interruzione del collegamento, o 1, cioè continuità di collegamento.

Ricordiamo che la codifica per un codice ciclico avviene sfruttando la matrice generatrice $\boldsymbol{G}$, come riportato nella (6.20)

$$(u_1 u_2 \dots u_k)\, \boldsymbol{G} = (c_{n-1} c_{n-2} \dots c_0)$$

Se il codice è assegnato in forma sistematica le prime k cifre di $\boldsymbol{x}$ sono d'informazione, mentre le restanti $n-k$ sono di controllo e vanno costruite sulla base delle prime k cifre. Poiché $\boldsymbol{c} = (c_{n-1} c_{n-2} \dots c_0)$ è parola di codice se e solo se la sua sindrome è nulla, assegnate $c_{n-1} c_{n-2} \dots c_{n-k}$ dovremo individuare le restanti $c_{n-k-1} c_{n-k-2} \dots c_0$ che portano all'annullamneto della sindrome.

Un modo equivalente di procedere è quello di considerare il codice ciclico definito mediante le sue radici $\beta, \beta^2, \dots, \beta^{d-1}$ e il suo polinomio generatore $g(x)$ dato dal prodotto dei polinomi minimi di $\beta, \beta^2, \dots, \beta^{d-1}$. Definito allora $c(x) = c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \dots + c_0$ si ha

$$\boldsymbol{c} \in \mathfrak{C} \Longleftrightarrow c(\beta) = 0, c(\beta^2) = 0, \dots, c(\beta^{d-1}) = 0 \tag{6.71}$$

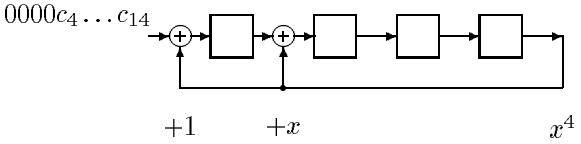
Assegnati $c_{n-1} c_{n-2} \dots c_{n-k}$ dobbiamo dunque individuare i restanti c_{n-k-1} $c_{n-k-2} \dots c_0$ in modo che valga la (6.71). Un metodo che porta alla soluzione è quello di dividere il polinomio $c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \dots + c_{n-k} x^{n-k}$ per $g(x)$

$$\sum_{i=n-k}^{n-1} c_i x^i = g(x) q(x) + r(x) \quad \text{con} \quad \text{gr}[r(x)] \leq n-k-1$$

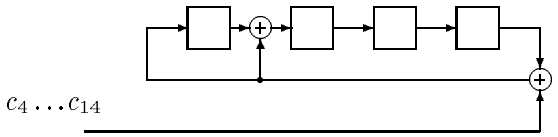
Ma allora

$$c(x) = \sum_{i=n-k}^{n-1} c_i x^i + r(x)$$

e il resto $r(x)$ che stiamo cercando è costituito dalle cifre di controllo. Esso viene individuato mandando in ingresso a un registro divisore la successione $c_{n-1}c_{n-2}\ldots c_{n-k}00\ldots0$, con $n-k$ zeri terminali. Effettuata la divisione si ha a disposizione $r(x)$, che specifica le cifre di controllo. Il circuito che segue illustra un esempio concreto col codice di Hamming C(15,11), associato al polinomio minimo $x^4 + x = 1$. Uno svantaggio del circuito è dato dal fatto che



bisogna attendere l'ingresso di tutti gli $n-k$ zeri prima di avere a disposizione le cifre di controllo come resto della divisione per $m_1(x)$. Per ovviare a tale inconveniente si può osservare che la rete di retroazione non interviene per i primi $n-k$ passi, cioè fin quando il primo coefficiente c_{n-1} giunge all'ultima cella. Si può allora inserire la sequenza direttamente in corrispondenza dell'ultima cella, realizzando in pratica una premoltiplicazione per x^{n-k}, così come evidenziato nella figura sottostante. Ciò consente di disporre del resto prima che gli zeri



entrino nel circuito, cioè non appena ha fatto il suo ingresso il termine c_{n-k}.

Lo schema completo di un codificatore per un codice ciclico è riportato in figura 6.6. Quando la sorgente emette i primi k simboli d'informazione $c_{n-1}c_{n-2}\ldots c_{n-k}$ i commutatori sono tutti nella posizione S; i simboli vanno direttamente sul canale e il registro comincia a effettuare il computo del resto, cioè delle $n-k$ cifre di controllo. Quando anche c_{n-k} è entrato nel canale, il registro ha già calcolato il resto e i commutatori vengono spostati sull'altra posizione; ciò determina l'uscita delle cifre di controllo dal registro e il contemporaneo azzeramento delle memorie, che costituisce lo stato iniziale per la prossima elaborazione. Il registro impiegato ha un numero di celle pari al grado di $g(x)$. Si osservi che il codificatore può essere costruito anche facendo ricorso ai *RLSR* di tipo 1; in questo caso il registro ha un numero di celle pari al grado del polinomio di controllo $h(x)$ e si sceglierà l'una o l'altra delle realizzazioni a seconda che sia $\mathrm{gr}[g(x)] < \mathrm{gr}[h(x)]$ o viceversa. Per questo tipo di circuiti rimandiamo a [9] o [59].

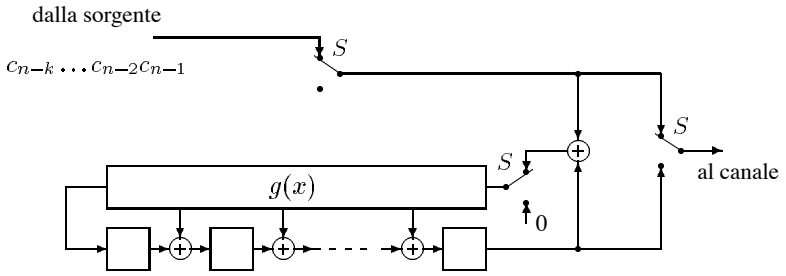


Figura 6.6 Circuito per la codifica cablata di un codice ciclico.

6.5.2 Circuiti di decodifica per un codice ciclico

Anche la decodifica può essere realizzata in modo semplice usando i registri a scorrimento retroazionato. Sempre supponendo di avere a disposizione un codice ciclico con radici $\beta, \beta^2, \dots, \beta^{d-1}$, ricevuto dal canale il vettore $\boldsymbol{v} = v_0 v_1 \dots v_{n-1}$ bisogna calcolare la sindrome $\boldsymbol{H}\boldsymbol{v}^T$, e verificare se è nulla. Facendo riferimento al polinomio associato $v_0 + v_1 x + \dots + v_{n-1} x^{n-1}$, deve essere

$$\boldsymbol{v} \in \mathfrak{C} \Longleftrightarrow v(\beta) = 0, v(\beta^2) = 0, \dots, v(\beta^{d-1}) = 0$$

Se la sindrome non è nulla bisogna intervenire per correggere l'errore sulla base delle informazioni che la sindrome stessa fornisce. Per semplicità illustriamo il procedimento facendo nuovamente riferimento al codice di Hamming $C(15, 11)$ associato al polinomio minimo $m_1(x) = x^4 + x + 1$ di grado $m = 4$ di cui α è radice, cioè $m_1(\alpha) = 0$. Ricordiamo inoltre che la matrice di controllo è

$$\boldsymbol{H} = (\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{-j}, \dots \alpha^{13} = \alpha^{-2}, \alpha^{14} = \alpha^{-1})$$

Si tratta allora di calcolare la sindrome $s(\boldsymbol{v}) = \boldsymbol{H}\boldsymbol{v}^T = v(\alpha)$ che sarà pari a α^{i-1} se l'errore è in posizione i. Dividendo $v(x)$ per $m_1(x)$ si ottiene

$$v(x) = m_1(x) q(x) + r(x) \;\text{ con }\; \text{gr}[r(x)] \leq m - 1$$

Calcolando il polinomio nella radice α si ha

$$s(\boldsymbol{v}) = v(\alpha) = r(\alpha) \;\text{ poiché }\; m_1(\alpha) = 0$$

Anche la sindrome si ricava come resto della divisione tra due polinomi, precisamente $v(x)$ e $m_1(x)$. Quando anche v_0 è entrato nel registro, la sindrome si

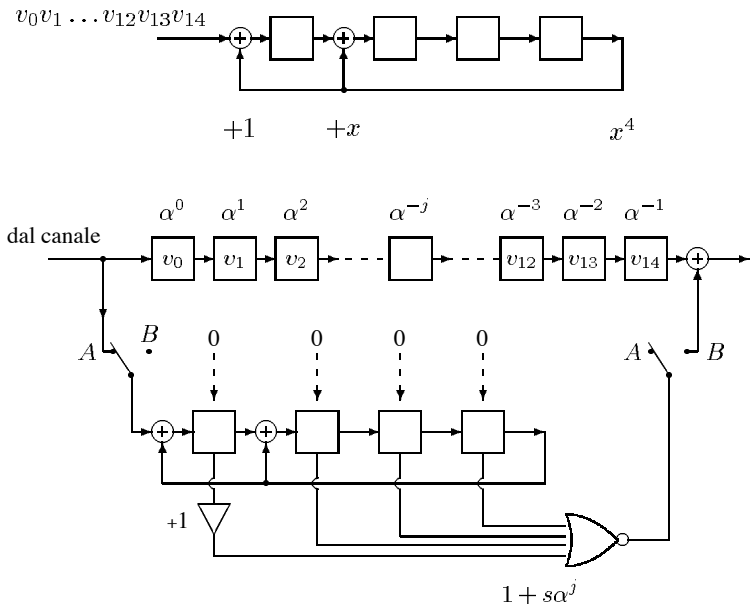


Figura 6.7 Esempio di circuito per la decodifica cablata di un codice ciclico.

legge nelle celle di memoria. La correzione avviene solitamente caricando la parola su un registro tampone e intervenendo direttamente sul bit errato non appena quest'ultimo lascia il registro. La figura 6.7 ci illustra un esempio del decodificatore completo. La parola di codice $v_0v_1 \ldots v_{14}$, che giunge dal canale affetta da un errore in posizione $15-j$ corrisponente alla colonna α^{-j}, entra nel registro tampone e nel registro $1 + x + x^4$ per il calcolo della sindrome; i commutatori sono dunque nella posizione A. Quando anche v_0 è entrato nel tampone si ha a disposizione la sindrome $s(\boldsymbol{v}) = s$, che corrisponde alla colonna α^{-j} della matrice di controllo $\boldsymbol{H}$. A questo punto i commutatori vengono posti in posizione B e si procede alla correzione nel modo seguente. Il registro divisore contenente la sindrome $s = \alpha^{-j}$ viene fatto evolvere autonomamente, eseguendo a ogni passo una moltiplicazione del suo contenuto per α. La successione degli stati è dunque $s, s\alpha, s\alpha^2, s\alpha^3 \ldots s\alpha^i \ldots$ mentre all'ingresso della porta NOR, per effetto della complementazione del bit di sinistra, si presenta l'elemento $1+s\alpha^i$. Se non ci sono stati errori la sindrome è nulla e lo è anche la successione vista sopra, l'ingresso della NOR è sempre a 1, la sua uscita è a 0 e non ci sono segnali di correzione. Se c'è invece errore in posizione $15-j$ la sindrome vale α^{-j} e si ha $1 + s\alpha^i = 0$ se e solo se $i = j$. Di conseguenza quando $i \neq j$ l'ingresso alla porta NOR è sempre diverso da 0 e l'uscita della porta è a 0. Viceversa,

quando dopo j passi si ha $1 + s\alpha^j = 0$ l'uscita della porta va a 1 e interviene un segnale che corregge il bit in corrispondenza di α^{-j} che in quel momento sta uscendo dal registro tampone. Quando anche l'ultimo bit ha lasciato la memoria tampone il registro divisore viene azzerato, il commutatore riposto in posizione A e un nuovo ciclo comincia per la parola successiva.

Estendiamo ora l'esempio al caso del codice BCH 2-correttore studiato nella sezione 6.3.4, la cui matrice di controllo è

$$\boldsymbol{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{-i} & \dots & \alpha^{-j} & \dots & \alpha^{-3} & \alpha^{-2} & \alpha^{-1} \\ 1 & \alpha^3 & \alpha^6 & \dots & \alpha^{-3i} & \dots & \alpha^{-3j} & \dots & \alpha^{-9} & \alpha^{-6} & \alpha^{-3} \end{pmatrix}$$

Se $\boldsymbol{v}$ è il vettore ricevuto, con due errori in corrispondenza delle colonne α^{-i} e α^{-j}, la sua sindrome è pari a

$$s(\boldsymbol{v}) = \begin{pmatrix} \boldsymbol{s_1} \\ \boldsymbol{s_3} \end{pmatrix} = \begin{pmatrix} \alpha^{-i} + \alpha^{-j} \\ \alpha^{-3i} + \alpha^{-3j} \end{pmatrix} = \begin{pmatrix} v(\alpha) \\ v(\alpha^3) \end{pmatrix} \tag{6.72}$$

Se $m_1(x) = x^4 + x + 1$ è il polinomio che ha α come radice, per effettuare la decodifica con la tecnica vista nel caso del codice di Hamming dobbiamo trovare $m_3(x)$ tale che sia $m_3(\alpha^3) = 0$, cioè

$$\boldsymbol{v} \in \mathfrak{C} \qquad \Longleftrightarrow \qquad \begin{matrix} m_1(\alpha) = 0 \\ m_3(\alpha^3) = 0 \end{matrix}$$

Dalla tabella 6.4 notiamo che il polinomio minimo di α^3 è $m_3(x) = x^4 + x^3 + x^2 + x = 1$, che è irriducibile. Ne risulta che

$$\begin{aligned} v(x) &= m_1(x)q_1(x) + r_1(x) \ \text{ con } \ \boldsymbol{s_1} = v(\alpha) = r_1(\alpha) \\ v(x) &= m_3(x)q_3(x) + r_3(x) \ \text{ con } \ \boldsymbol{s_3} = v(\alpha^3) = r_3(\alpha^3) \end{aligned}$$

Anche per il calcolo della sindrome $\boldsymbol{s_3}$ possiamo usare un circuito divisore; in questo caso bisogna dividere per $m_3(x) = x^4 + x^3 + x^2 + x = 1$, tenendo però conto che il resto va calcolato in α^3. Se $r_3(x) = a + bx + cx^2 + dx^3$ si ha $r_3(\alpha^3) = a + b\alpha^3 + c\alpha^6 + d\alpha^9$, con α^6 e α^9 che vanno ridotti tenendo conto che $\alpha^4 = \alpha + 1$. Sostituendo si ottiene $r_3(\alpha^3) = a + d\alpha + c\alpha^2 + (b + c + d)\alpha^3$.

Per effettuare la correzione sincrona dei due errori, secondo il principio visto nel caso del codice di Hamming, riprendiamo l'equazione $1 + \boldsymbol{s}\alpha^i = 0$ sulla quale tale correzione si basa. Quando $\boldsymbol{s} = \alpha^{-j}$ essa può essere riscritta come

$$1 + \alpha^{-j}\boldsymbol{z} = 0 \tag{6.73}$$

e interpretata nel modo seguente: se $\boldsymbol{s} = \alpha^{-j}$ è la sindrome, la correzione interverrà sul bit che esce dal registro tampone dopo j passi, cioè quando $\boldsymbol{z} = \alpha^j$. L'incognita $\boldsymbol{z}$ rappresenta dunque il *reciproco della posizione d'errore*.

La (6.73) è un caso particolare dell'equazione (6.42) per un codice 1-correttore, con l'unica differenza che la radice corrisponde in questo caso al reciproco della posizione d'errore. L'equazione associata al polinomio locatore (6.41) può essere riscritta con le sostituzioni $\boldsymbol{u} = \alpha^{-i}$ e $\boldsymbol{v} = \alpha^{-j}$, che portano all'equazione (in $GF(2)$)

$$\sigma(\boldsymbol{z}) = 1 + \sigma_1 \boldsymbol{z} + \sigma_2 \boldsymbol{z}^2 = \left(1 + \boldsymbol{z}\alpha^{-i}\right)\left(1 + \boldsymbol{z}\alpha^{-j}\right) = 0$$
$$\text{con} \quad \begin{array}{l} \sigma_1 = \alpha^{-i} + \alpha^{-j} = \boldsymbol{s_1} \\ \sigma_2 = \alpha^{-i}\alpha^{-j} = \boldsymbol{s_1}^2 + \boldsymbol{s_3}\boldsymbol{s_1}^{-1} \end{array} \tag{6.74}$$

le cui radici rappresentano *il reciproco delle posizioni d'errore*. L'equazione (6.74) è formalmente simile alla (6.73) e come questa viene risolta in modo indiretto mediante una semplice verifica sistematica di tutte le radici. In altre parole, avute le due sindromi $\boldsymbol{s_1}$ e $\boldsymbol{s_3}$ si calcolano σ_1 e σ_2 dalla (6.74). Nel momento in cui il bit in posizione α^{-h} lascia la memoria tampone dopo h passi, si verifica se $\sigma(\boldsymbol{\alpha^h}) \neq 0$; se ciò accade il segnale di correzione dev'essere nullo; viceversa, quando $h = j$ (oppure $h = i$) si ha $\sigma(\boldsymbol{\alpha^j}) = 0$ e si deve mandare un segnale di correzione.

I circuiti per la decodifica cablata richiedono, oltre ai divisori per il calcolo delle due sindromi, due ulteriori registri che effettuino il prodotto rispettivamente per α e per α^2 (per il primo si può in realtà continuare a usare il registro divisore, così come si è fatto nel caso del codice di Hamming). In tal modo non appena l'ultimo bit è entrato nella memoria tampone, nei registri divisori sono disponibili le sindromi, e tramite queste i coefficienti σ_1 e σ_2, che possono essere ricavati direttamente da un circuito cablato accessorio. Tali coefficienti costituiscono lo stato iniziale dei due registri moltiplicatori per α e α^2 e dopo il primo ciclo l'ingresso della porta NOR diventa $\sigma(\alpha) = 1 + \sigma_1\alpha + \sigma_2\alpha^2$ e si verifica se α è radice; se ciò accade all'uscita di v_{14} viene mandato il segnale di correzione altrimenti dalla porta NOR esce 0. Al passo successivo l'ingresso NOR vale $\sigma(\alpha^2) = 1 + \sigma_1\alpha^2 + \sigma_2\alpha^4$ e si ripete il procedimento per ciascun $\sigma(\alpha^h) = 1 + \sigma_1\alpha^h + \sigma_2\alpha^{2h}$ $(1 \leq h \leq 15)$.

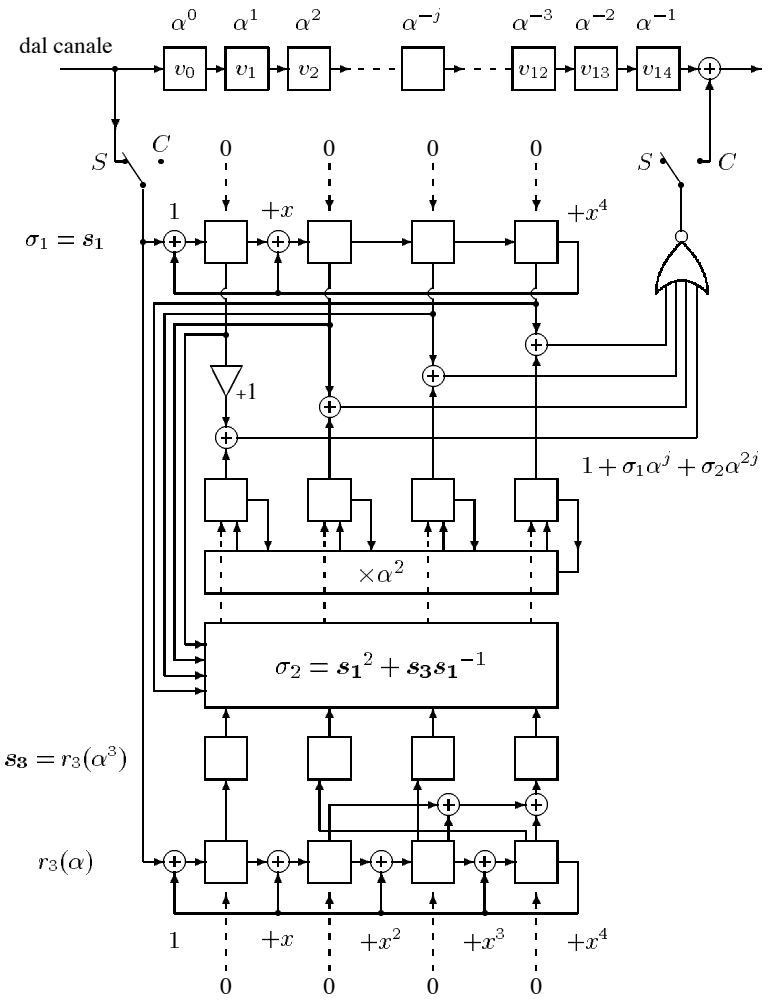


Figura 6.8 Circuito completo per la decodifica cablata di un codice BCH(15,7).

6.6 Limitazioni normative

Considereremo ora il problema di individuare le prestazioni asintotiche dei codici rispetto ai due parametri più importanti, cioè il tasso R del codice e la sua capacità λ di correzione degli errori. È evidente che le due esigenze perseguite, alta capacità di correzione e tasso elevato, sono in conflitto tra loro; dal punto di vista attuativo si tratterà di trovare una soluzione di compromesso, che consenta di trasmettere informazione con un tasso adeguato e confidando in una capacità di correzione asintoticamente discosta dallo zero. La situazione di compromesso viene modulata agendo sulla cardinalità del dizionario e sulla posizione reciproca delle parole di codice.

Intuitivamente, fissando a priori uno dei due parametri si possono imporre delle limitazioni (superiori o inferiori) per l'altro. Nel seguito studieremo il comportamento asintotico di tali limitazioni quando $n \to \infty$, delimitando nel diagramma λ, R la zona che risulta compatibile con l'esistenza di codici effettivi.

Siano

$q =$ cardinalità dell'alfabeto di canale;

$n =$ lunghezza delle parole di codice;

$M = |C| =$ numero di n-ple che sono parole di codice;

$d = d_{min} =$ distanza minima del codice;

$\lambda = d/n =$ tasso di correzione del codice;

$R = \log_q M/n =$ tasso del codice misurato in q-it.

i parametri di un codice correttore.

Def. 6.9. *Fissato d sia $M^* \triangleq \max\{M : \text{esiste un C(n,M,d) codice}\}$. Un codice $C : |C| = M^*$ è detto ottimo.*

Fissati dunque q, n, λ andiamo a studiare il comportamento asintotico del tasso ottimo $R^*(q, n, \lambda)$.

6.6.1 Limitazione di Singleton

La limitazione di Singleton (e le due che seguiranno immediatamente, Plotkin e Hamming) sono delle *limitazioni superiori*; esse ci indicano ciò che *non* è possibile ottenere da qualunque codice in termini di R fissato λ o viceversa.

Con i parametri di cui sopra, supponiamo di ordinare tutte le parole di un certo codice disponendole l'una sopra l'altra, evidenziando per ciascuna di esse un prefisso di lunghezza $n - d + 1$ e un suffisso di lunghezza $d - 1$. L'idea è quella di togliere, da ciascuna parola, l'eccesso d'informazione che serve a differenziarla da tutte altre (il suffisso), mantenendo così la diversificazione al

minimo livello possibile (cioè $d_{min} = 1$). Se togliamo le $d-1$ lettere del suffisso passiamo da un codice $C(n, M, d)$ a un codice $C(n-d+1, M, 1)$, avendo dunque garantita una $d_{min} \geq 1$ sufficiente per distinguere tutte le parole.

Esempio 6.28. Sia $M = 4, G = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$ e $\mathcal{C} = \{00000, 01110, 11011, 10101\}$. La distanza minima vale $d_{min} = 3$. Se togliamo le ultime due lettere la distanza minima diventa pari a 2

$$\begin{array}{ccc|cc} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{array} \qquad \begin{array}{c|cc|cc} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{array}$$

Con un'altra scelta, per esempio eliminando la 2 e 3 colonna, si ottiene invece un codice con $d_{min} = 1$.◯

A questo punto è facile trovare una limitazione superiore per le parole del codice (e quindi, implicitamente, per il tasso) poiché il numero di prefissi non può superare il massimo numero di stringhe q-arie di lunghezza pari alla lunghezza del prefisso

$$M \leq q^{n-d+1}$$

Se ci si basa sul tasso si può scrivere

$$R = \frac{\log_q M}{n} \leq \frac{1}{n} \log_q q^{n-d+1} = 1 - \lambda + \frac{1}{n}$$

Se il codice che stiamo esaminando è un codice lineare $C(n, k)$, la relazione diventa $k \leq n - d + 1$. Se la limitazione vale come ugualianza il codice si dice *MDS* (*maximum distance separable*).

La forma asintotica della limitazione di Singleton diventa dunque

$$R \leq 1 - \lambda$$

Fissato λ il tasso non può dunque stare al di sopra della retta passante per i punti (0,1) e (1,0) che delimita così la zona permessa (si veda la figura 6.9).

6.6.2 Limitazione di Plotkin

Anche in questo caso si fornisce una limitazione superiore, che si ricava considerando la cosiddetta *distanza globale* del codice, definita come la somma delle distanze di tutte le possibili coppie di parole di codice. Poiché c'è una distanza minima garantita d_{min}, la distanza globale può essere limitata inferiormente non

appena si consideri che tra ciascuna delle possibili $M(M-1)$ coppie ordinate di parole la distanza deve essere pari almeno a d_{min}. Si ha dunque

$$D \triangleq \sum_{\substack{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}\\ \boldsymbol{x}\neq\boldsymbol{y}}} d_H(\boldsymbol{x},\boldsymbol{y}) \;\geq\; M(M-1)\,d$$

si noti che secondo tale definizione ciascuna coppia viene contata due volte; per i nostri scopi ciò non risulta però pregiudizievole.

Cerchiamo ora una limitazione superiore per D, per poterla poi confrontare con quella inferiore. A tal scopo computiamo la distanza globale sulla base delle colonne, invece che sulle righe com'è stato appena fatto. Si supponga di disporre le n-ple una sull'altra, in modo da poter evidenziare il contributo che ciascuna colonna porta alla distanza globale. Si ottiene

$$D \;=\; \sum_{i=1}^{n}\sum_{j=1}^{q} M_{ij}\,(M-M_{ij}) = \sum_{i=1}^{n}\sum_{j=1}^{q}\left(M_{ij}M - M_{ij}^2\right) = \\ = nM^2 \;-\; \sum_{i=1}^{n}\sum_{j=1}^{q} M_{ij}^2$$

dove M_{ij} rappresenta il numero di lettere j-esime nella colonna i, mentre $M - M_{ij}$ il numero di simboli nella colonna i-esima che sono diversi dal simbolo j. La terza uguaglianza deriva infine dal fatto che $\sum_{j=1}^{q} M_{ij} = M$. Anche in questo caso ciascuna coppia (di lettere) è stata contata due volte. Si osservi ora che

$$\sum_{j=1}^{q} M_{ij}^2 = \left(\frac{1}{q}\sum_{j=1}^{q} 1^2\right)\left(\sum_{j=1}^{q} M_{ij}^2\right) \;\geq \frac{1}{q}\left(\sum_{j=1}^{q} 1\cdot M_{ij}\right)^2 \;=\; \frac{M^2}{q}$$

La limitazione superiore per D diventa

$$D \leq n\,M^2 - \sum_{i=1}^{n}\frac{M^2}{q} \;=\; n\,M^2\,\frac{q-1}{q}$$

Dal confronto tra la limitazione superiore e quella inferiore per la distanza globale si ottiene $M\;(M-1)d \;\leq\; n\,M^2\;(q-1)/q = n\,M^2\,\theta$, con $\theta = \frac{q-1}{q}$, cioè

$$M \;\leq\; \frac{d}{d-\theta n} \qquad \text{se} \qquad d > \theta n \tag{6.75}$$

oppure

$$\lambda \leq\; \frac{M}{M-1}\,\frac{q-1}{q} \tag{6.76}$$

che è la limitazione cercata. Assegnato un certo tasso (cioè un valore di M poiché $R = 1/n \log_q M$ e $M = q^{nR}$) la capacità di correzione non può dunque superare il valore specificato dalla (6.76). Il comportamento asintotico della limitazione lo si ricava sostituendo M con q^{nR} e facendo tendere n all'infinito.

$$\lambda \leq \frac{q-1}{q} \frac{1}{1-\frac{1}{q^{nR}}} \stackrel{n\to\infty}{\longrightarrow} \frac{q-1}{q}$$

che è la versione asintotica della limitazione di Plotkin. Sul diagramma di figura 6.9 si può vedere la zona permessa che deriva dalle limitazioni di Singleton e Plotkin nel caso in cui si abbia un alfabeto binario, cioè $q = 2$.

Oss. 6.7. La limitazione asintotica di Plotkin vale solo se $R > 0$ strettamente, e dunque anche il tratto dell'ascissa compreso tra 1/2 e 1 *può contenere dei codici* il cui tasso asintotico risulta nullo.

La versione asintotica testé illustrata può essere migliorata nel modo seguente: al posto del codice $C(n, M, d)$ si consideri un codice $C'(n', M', d)$ in cui sia $n' < n$ e venga mantenuta la stessa distanza minima d. Se M' è il numero delle parole di questo nuovo codice, dovrà essere $M \leq q^{n-n'}M'$, cioè $M' \geq q^{n-n'}M$. Se ora scegliamo $n' = \lfloor (d-1)/\theta \rfloor$ si ottiene $d - \theta n' \geq 1$, che sostituita nella (6.75) porta a $M' < d$. Dal confronto con la limitazione inferiore per M' si ottiene $M < dq^{n-n'}$. Facendo ora tendere n all'infinito e trascurando i termini infinitesimi si ottiene $n'/n \to \lambda/\theta$ e

$$R \leq 1 - \lambda/\theta \tag{6.77}$$

che per $q = 2$ porta alla retta che passa per i punti (1/2,0) e (0,1).

6.6.3 Limitazione di Hamming

La limitazione di Hamming è nota in letteratura anche col nome di *sphere packing bound*. Ciò deriva dal seguente ragionamento: poiché tra le parole di codice pensate immerse in uno spazio *n*-dimensionale esiste una distanza minima garantita d_{min}, si può immaginare di centrare una sfera di raggio ρ in ogni parola di codice, in modo tale che non si verifichino intersezioni con le sfere adiacenti. Ciò corrisponde ad un "impacchettamento" delle sfere. Ciascuna sfera contiene, nel suo volume, tutte le *n*-ple che si trovano a distanza di Hamming minore o uguale a ρ e che, ovviamente, non sono parole di codice. Da ciò si ricava che il prodotto tra M e il volume della singola sfera sarà limitato superiormente dal numero di possibili *n*-ple sull'alfabeto *q*-ario. Ricordiamo che il volume di una sfera di Hamming si ricava sommando le cardinalità dei singoli

gusci a distanza i dal centro della sfera, secondo quanto riportato dalla relazione (4.36)

$$\text{Vol}[S_\rho(c)] \;=\; \sum_{i=0}^{\rho} \binom{n}{i} (q-1)^i$$

Se ora d è la distanza minima e il codice corregge tutte le configurazioni di t errori, deve essere $d \geq 2t+1$, e dunque sceglieremo ρ pari a $\lfloor (d-1)/2 \rfloor$, in modo da garantire che le sfere siano disgiunte. Ma il prodotto del numero di sfere, cioè di parole di codice, per il volume di ciascuna sfera deve essere limitato da q^n

$$M \left[\sum_{i=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{i} (q-1)^i \right] \leq \; q^n$$

Prendendo infine il logaritmo e dividendo per n si ottiene la *limitazione di Hamming al finito*

$$R \;\leq\; 1 \;-\; \frac{1}{n} \log_q \left[\sum_{i=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{i} (q-1)^i \right] \tag{6.78}$$

Il risultatato non è molto leggibile a causa della presenza del coefficiente binomiale. Possiamo però ricorrere alle limitazioni offerte dalla teoria dei tipi esatti, che ci consentono di esprimere un'approssimazione asintotica degli stessi coefficienti. La formula (3.44), tenendo conto della (3.33) calcolata per $k = 2$, restituisce

$$(n+1)^{-1}\, q^{n\, h_q(\frac{i}{n})} \leq \binom{n}{i} \leq q^{n\, h_q(\frac{i}{n})}$$

con $h_q(x) \;=\; -x \, \log_q x - (1-x) \, \log_q (1-x)$. Moltiplicando tutti i membri per $(q-1)^i$ si ottiene

$$(n+1)^{-1}\, q^{n\, [h_q(\frac{i}{n}) + \frac{i}{n} \log_q (q-1)]} \leq \binom{n}{i} (q-1)^i \leq q^{n\, [h_q(\frac{i}{n}) + \frac{i}{n} \log_q (q-1)]}$$

La struttura dell'esponente suggerisce di ricorrere all'entropia q-aria $H_q(\epsilon) = h_q(\epsilon) + \epsilon \log_q (q-1)$, con $0 \leq \epsilon \leq \frac{q-1}{q}$ (si veda la (4.17)). Sostituendo $H_q(\frac{i}{n})$ e sommando su *i* fino al raggio ρ si ottiene

$$(n+1)^{-1} \sum_{i=0}^{\rho} q^{n\, H_q(\frac{i}{n})} \leq \sum_{i=0}^{\rho} \binom{n}{i} (q-1)^i \leq \sum_{i=0}^{\rho} q^{n\, H_q(\frac{i}{n})}$$

Poiché $H_q(\frac{i}{n})$ è crescente con i possiamo conservare a sinistra solo $H_q(\frac{\rho}{n})$ e limitare a destra con $(n+1)\ H_q(\frac{\rho}{n})$

$$(n+1)^{-1}\ q^{n\ H_q(\frac{\rho}{n})} \leq \sum_{i=0}^{\rho} \binom{n}{i}(q-1)^i \leq (n+1)q^{n\ H_q(\frac{\rho}{n})}$$

Facendo il logaritmo e dividendo per n si ottiene

$$\begin{aligned} H_q(\frac{\rho}{n}) - \frac{1}{n}\log_q(n+1) &\leq \frac{1}{n}\log_q\left[\sum_{i=0}^{\rho}\binom{n}{i}(q-1)^i\right] \\ &\leq H_q(\frac{\rho}{n}) + \frac{1}{n}\log_q(n+1) \end{aligned} \tag{6.79}$$

Ora, quando n tende all'infinito si ha $\frac{1}{n}\log_q\left[\sum_{i=0}^{\rho}\binom{n}{i}(q-1)^i\right] \to H_q(\frac{\rho}{n})$, che sostituita nell'espressione (6.78) della limitazione di Hamming al finito e tenuto conto che $\frac{\rho}{n} = \frac{\lfloor (d-1)/2 \rfloor}{n} \overset{n\to\infty}{\longrightarrow} \frac{d}{2n} = \frac{\lambda}{2}$ conduce alla versione asintotica della limitazione di Hamming

$$R \ \leq \ 1 \ - \ H_q(\frac{\lambda}{2}) \tag{6.80}$$

il cui diagramma è riportato nella figura 6.9.

La migliore limitazione disponibile al momento, che risale comunque al '77, è dovuta a McEliece, Rodemich, Rumsey and Welch [63], che nel caso binario è data dalla relazione

$$R \ \leq \ H_2\left(\frac{1}{2} - \sqrt{\lambda(1-\lambda)}\right) \tag{6.81}$$

(si veda il diagramma di figura 6.9), per la cui dimostrazione rimandiamo a [63]

6.6.4 Limitazione di Gilbert-Varšamov

È questo il primo esempio di limitazione inferiore, che assicura l'esistenza di (almeno) un codice che possiede delle prestazioni minime garantite, in termini di tasso di trasmissione e di capacità di correzione degli errori. Il ragionamento sotteso fa sempre riferimento alle sfere di Hamming, e anche in questo caso si tratta di "riempire" lo spazio delle *n*-ple in modo opportuno. Purtroppo la procedura, pur essendo costruttiva in senso stretto, individua in genere dei codici che sono privi di una qualche struttura adatta a una loro gestione efficiente delle operazioni di codifica/decodifica. L'algoritmo sul quale si basa la limitazione di Gilbert porta dunque, come risultato finale, a un codice funzionale alla sola limitazione, il che naturalmente è tutt'altro che trascurabile dal punto di vista normativo.

Se dunque $d = d_{min} \geq 2t + 1$ è la distanza minima del codice in grado di correggere tutte le configurazioni fino a t errori, consideriamo un'n-pla qualunque, p.es $\boldsymbol{x}$, e centriamo su di essa una sfera di raggio $d - 1$. A questo punto verifichiamo se il numero di *n*-ple comprese nella stessa, cioè il suo volume, è maggiore del numero totale di *n*-ple. Se ciò non accade si ha

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

e possiamo prendere un'altra *n*-pla $\boldsymbol{y}$, che non appartiene alla sfera centrata in $\boldsymbol{x}$, la cui esistenza è garantita dal fatto che il volume della sfera non satura ancora tutto lo spazio. Controlliamo nuovamente se il volume totale delle sfere è superiore al numero totale di *n*-ple e iteriamo la procedura finché si trova M tale che

$$M \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i \ \geq \ q^n$$

cioè

$$R \ \geq \ 1 \ - \ \frac{1}{n} \log_q \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i \tag{6.82}$$

che è la versione al finito della limitazione cercata. Abbiamo così costruito un codice la cui distanza minima è d e il cui tasso è limitato inferiormente dalla (6.82).

Oss. 6.8. La (6.82), che porta in genere a un codice codice non lineare, può essere facilmente migliorata eliminando dal volume delle sfere che via via si aggiungono nel procedimento appena descritto il contributo che deriva *dall'intersezione* tra le sfere (si veda [28]). Il miglioramento è tuttavia asintoticamente trascurabile.

Lo studio asintotico è identico a quello fatto per la limitazione precedente, con l'unica differenza che il raggio delle sfere è $d - 1$ invece che $\lfloor (d-1)/2 \rfloor$. La versione asintotica della limitazione di Gilbert risulta essere allora

$$R \ \geq \ 1 \ - \ H_q(\lambda) \tag{6.83}$$

Mentre la versione asintotica della limitazione superiore è stata progressivamente affinata nel corso degli anni (le limitazioni di Singleton, Plotkin e Hamming studiate in questa sede sono solo le più significative), per la versione asintotica della limitazione inferiore, subito dopo i lavori di Gilbert e di Varšamov che risalgono agli anni '50, non ci sono stati miglioramenti di alcun tipo. Solo nel 1982 Tsfasman, S.G. Vlăduţ e T. Zink riuscirono a trovare una limitazione migliore di quella di Gilbert-Varšamov (*GV*), che vale però "solamente" per $q \geq 49$. A

questo punto si congettura che la limitazione *GV* nel caso binario sia stretta, e quindi non ulteriormente migliorabile.

Il discorso cambia molto se si passa al finito, dove la limitazione (6.82) è stata superata in molte occasioni, soprattutto per classi speciali di codici. Un esempio molto semplice lo si ha quando si considerano i codici lineari; in questo caso la costruzione della matrice di controllo $\boldsymbol{H}$ si può effettuare con la seguente procedura:

1. si scelga a caso una prima colonna diversa dalla colonna nulla
2. si scelga a caso una seconda colonna (non nulla) non proporzionale alla prima
3. in generale si scelga l'i-esima colonna in modo tale che non sia combinazione lineare di $d-2$ (o meno) colonne tra quelle scelte precedentemente (ciò garantisce una distanza minima pari a d).

Il passo 3) assicura che non ci sono sottinsiemi di $d-1$ o meno colonne che sono linearmente dipendenti, ed esso può essere iterato fin quando il numero di possibili combinazioni lineari di $d-2$ (o meno) colonne non supera q^{n-k}, il numero totale di colonne. Ciò determina una limitazione inferiore analoga alla (6.82), ma più stertta poiché si ha $d-2$ al posto di $d-1$

$$\sum_{i=0}^{d-2} \binom{n}{i} (q-1)^i \geq q^{n-k} \tag{6.84}$$

Altri esempi di miglioramenti, oltre a quello relativo all'osservazione 6.8, sono forniti sono illustrati nei lavori di Hashim [43], Elia [26]) e Barg [4].

I diagrammi della figura 6.9 illustrano le varie limitazioni asintotiche studiate, evidenziando con un tratteggio l'area nella quale *possono* esistere dei codici. Da questo punto di vista l'unica garanzia che abbiamo è che ne esistono sopra la linea della limitazione inferiore di Gilbert-Varšamov. Per quanto riguarda le limitazioni superiori si osservi che quella di McEliece-Rodemich-Rumsey-Welch è la migliore solo a partire da $\lambda = 0.1497$; per valori inferiori prevale invece la limitazione di Elias .

La figura mette anche in evidenza le pessime prestazioni asintotiche dei codici precedentemente studiati. Si ricordi inoltre che anche l'intervallo chiuso [0.5-1] è compatibile con l'esistenza di codici. Infatti quelli a ripetizione occupano la posizione estrema di $\lambda = 1$, mentre i codici di Hadamard si trovano n corrispondenza del valore 1/2.

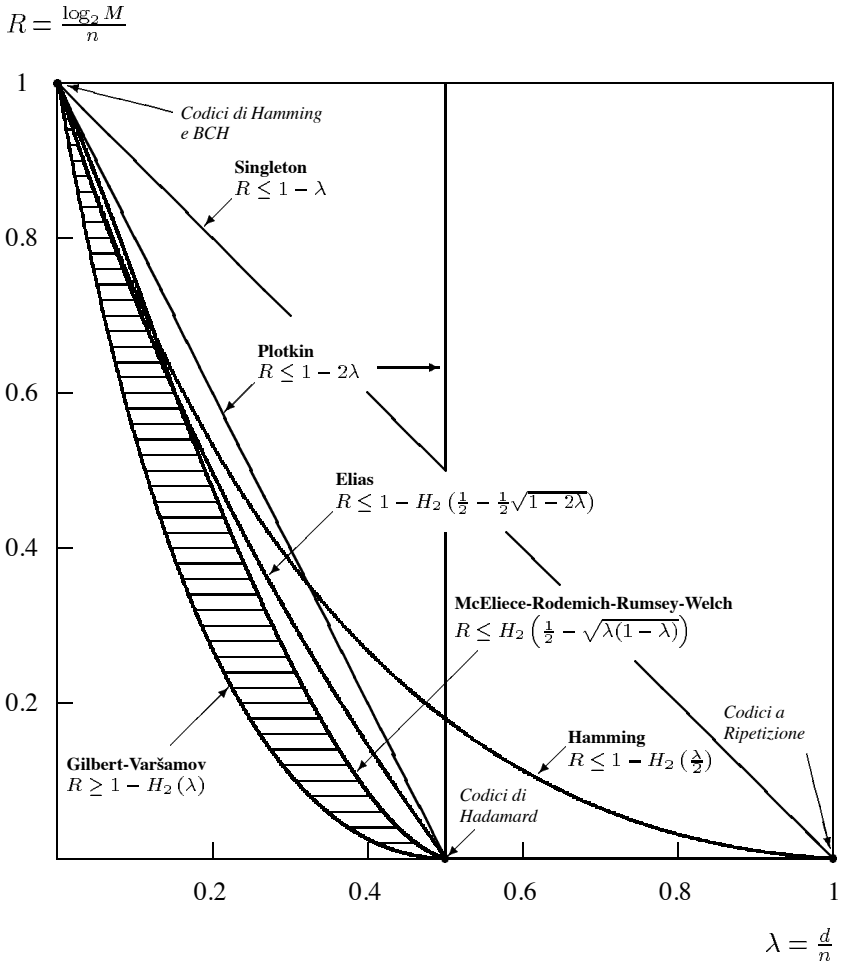


Figura 6.9 Diagramma delle limitazioni asintotiche.

Parte III

Cifrari

Capitolo 7
Introduzione alla Crittologia

La Crittologia, nella sua accezione moderna, è una disciplina che studia le tecniche matematiche necessarie a conseguire gli obiettivi di una trasmissione sicura di dati su canali normalmente accessibili anche a utenti non autorizzati alla loro acquisizione (che chiameremo, a seconda del contesto, oppositori, spie, frodatori, falsificatori ecc.). La sicurezza, in questo, caso non attiene dunque alla possibilità che l'informazione venga corrotta dal rumore del canale, ma è relativa all'ipotesi che un utente non autorizzato riesca a *intercettarla*, compromettendo così la *riservatezza* della trasmissione ed eventualmente anche *l'integrità* dei dati in transito.

D'altra parte se l'utente X deve indirizzare a un certo altro utente Y dei dati riservati, prima d'iniziare la trasmissione deve assicurarsi circa la reale *identità* di Y, che deve essere provata con una qualche procedura di *autenticazione*. Ma anche Y, ricevendo il messaggio riservato, deve avere garanzie sul fatto che esso provenga effettivamente da X e non da un falsificatore F che *impersona* X.

Anche quando le identità dei due utenti X e Y fossero state accertate, c'è però sempre la possibilità che il messaggio segreto spedito da X a Y possa esser stato compromesso da F, che può aver aggiunto, tolto o modificato qualche parte del messaggio a insaputa di X e Y (si noti per altro che questa operazione potrebbe anche essere fine a sé stessa, non avendo F la possibilità di *controllare* l'esito delle modificazioni apportate).

Inoltre l'utente X, dopo aver trasmesso un messaggio legittimo all'utente Y, potrebbe *disconoscere* tale messaggio, affermando che si tratta di un falso non a sé attribuibile.

Da questi semplici esempi si deduce che il contesto nel quale opera la Crittologia è molto variegato, e che i problemi connessi con una trasmissione sicura dei dati coinvolgono in realtà diversi livelli per i quali è necessario fornire delle garanzie specifiche. Si parte dunque dal livello dei dati, per i quali si devono garantire la riservatezza e l'integrità, per passare poi al livello delle identificazioni personali degli utenti, con tutti i giochi di impersonazione, falsificazione e

sostituzione possibili.

La crittologia è lo studio congiunto di *crittografia* e *crittanalisi* (vedi introduzione generale). La prima svolge un ruolo per così dire propositivo, nel senso che studia nuovi metodi per proteggere le informazioni o per garantire l'autenticità di un messaggio o la sua integrità, mentre alla crittanalisi spetta il compito di escogitare metodi per *forzare* gli schemi di cifratura proposti, rendendo vano il lavoro dei crittografi. Questo atteggiamento "distruttivo" della crittanalisi è tuttavia il presupposto necessario per poter affermare la sicurezza di un certo cifrario proposto. L'utente che intende acquisire un cifrario (o un protocollo d'identificazione) desidera infatti avere delle garanzie sul suo funzionamento; queste possono essere fornite dal costruttore per via diretta, *dimostrando* con un procedimento logico-matematico sotto quali condizioni il sistema è sicuro (o qual è il livello minimo di risorse computazionali necessarie per forzare il sistema), oppure più semplicemente per via indiretta, invitando la comunità scientifica a individuare un metodo per forzare il cifrario (o il protocollo) proposto.

La prima soluzione è per ovvie ragioni solitamente difficile da percorrere, soprattutto quando si richieda una dimostrazione di *sicurezza incondizionata* (per altro non sempre possibile), anche se in taluni casi si riesce a ricondurre la misura della sicurezza a una valutazione della complessità computazionale di un qualche problema noto. Tale soluzione è senza dubbio quella privilegiata.

La seconda possibilità è usata soprattutto nei casi in cui si voglia mantenere segreta la struttura del cifrario (o quantomeno le ragioni che hanno suggerito tale struttura!) oppure quando non si riesca a inquadrare matematicamente la sicurezza associata al sistema. Qualunque sia la scelta, il lavoro dei crittanalisti assume comunque un ruolo fondamentale.

I problemi accennati precedentemente, e che costituiscono il terreno di sviluppo delle tecniche crittografiche, sono essenzialmente riconducibili alle seguenti categorie:

Obiettivi di un sistema crittografico

1. Protezione della riservatezza dell'informazione.

2. Autenticazione dell'utente legittimo.

3. Verifica dell'integrità dei dati.

4. Non ripudiabilità di un utente che ha effettuato una trasmissione legittima.

Si possono però ricordare anche i problemi legati *al controllo degli accessi* a risorse di varia natura da parte di utenti con diversi livelli di privilegio (per esempio nei sistemi operativi), l'acquisizione congiunta di un certo livello di privilegio da parte di più utenti (X e Y hanno accesso a una risorsa solo quando la richiedono congiuntamente), la protezione nei confronti di attacchi di tipo fisico a danno di memorie, processori ecc. (tanto da parte di utenti non autorizzati, che hanno lo scopo di acquisire direttamente il sostrato fisico contenente le informazioni, quanto da parte di eventi naturali, tipo inquinamenti elettromagnetici, cataclismi ecc., che possono portare a una perdita delle informazioni).

In generale si può ricorrere anche a una distinzione tra *attacchi passivi*, cioè l'acquisizione pura e semplice di dati sensibili, e *attacchi attivi*, nei quali l'opponente s'introduce nel canale di trasmissione modificando i dati in transito (*attacco di sostituzione*), oppure impersonando la sorgente, cioè generando egli stesso un messaggio falsamente attribuito alla sorgente legittima (*attacco d'impersonazione*). Ecco allora che la segretezza è un attributo relativo agli attacchi di tipo passivo, mentre l'autenticazione della sorgente e la verifica dell'integrità dei dati sono contromisure da adottare per la prevenzione di attacchi attivi.

I nostri obiettivi sono limitati, nel senso che affronteremo i problemi solamente dal punto di vista dei principi generali, lasciando ai testi specializzati l'onere di consentire al lettore un approfondimento (teorico e tecnico-operativo) dei diversi schemi che analizzeremo. Tralasceremo inoltre la trattazione della sicurezza fisica dei dispositivi, poiché prescinde dagli obiettivi di questo testo. Rimandiamo dunque all'eccellente manuale *Handbook of Applied Cryptography* [64] per una rassegna su tutti gli aspetti della crittografia e, soprattutto, come preziosa fonte di referenze bibliografiche.

7.1 Impostazione generale

7.1.1 Segretezza

Per proteggere la segretezza delle informazioni in transito sul canale (o memorizzate su una memoria locale) nei confronti di utenti non autorizzati alla loro acquisizione è necessario effettuare una *cifratura* del testo, trasformando il *testo in chiaro* in un *testo cifrato* (o *crittogramma*); ciò corrisponde ad applicare una opportuna trasformazione, o *chiave*, scelta all'interno del cosiddetto *spazio delle chiavi* $\mathcal{K}$. La trasformazione deve essere *invertibile* per poter consentire la *decifrazione*, cioè il recupero del testo in chiaro a partire dal testo cifrato conoscendo la chiave usata. Se dunque chiamiamo rispettivamente $\mathcal{M}$ e $\mathcal{C}$ lo *spazio dei messaggi* e *lo spazio dei crittogrammi* si ha la seguente

Def. 7.1. *Un cifrario è una famiglia finita di trasformazioni invertibili, definite sullo spazio* $\mathcal{M}$ *dei messaggi e con valori nello spazio* $\mathcal{C}$ *dei crittogrammi.*

Disponendo del messaggio $m \in \mathcal{M}$ e della chiave $k \in \mathcal{K}$ possiamo effettuare cifratura e decifrazione nel modo seguente

$$\begin{array}{lll} m, k \longrightarrow & c = Cif(m,k) = C_k(m) & \text{Cifratura} \\ c, k \longrightarrow & m = Dec(c,k) = D_k(c) = D_k C_k(m) & \text{Decifrazione} \end{array} \tag{7.1}$$

dove $Cif(m,k) = C_k(m)$ e $Dec(c,k) = D_k(c)$ sono funzioni *deterministiche* delle loro variabili. La figura 7.1 illustra un esempio elementare di cifrario. Poichè abbiamo a che fare con una funzione invertibile, è ovvio che per ciascun messaggio m_i esiste un solo arco per ogni k_i scelta e che la stessa chiave non può trasformare due messaggi diversi nello stesso crittogramma. È inoltre ovvio

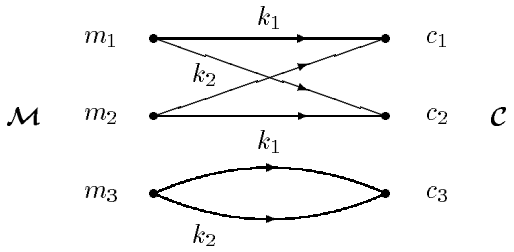


Figura 7.1 Un semplice esempio di cifrario.

che la chiave deve essere occultata a tutti gli utenti non autorizzati alla lettura delle informazioni riservate; inoltre, come evidenziato dalla (7.1), essa viene impiegata dall'utente legittimo Y per eseguire la decifrazione, e dunque l'utente Y e la sorgente X devono *concordare* una chiave. Questo comporta il problema di trasferire la chiave a distanza, oppure di far "incontrare" X e Y prima che i due si scambino le informazioni riservate, in modo che essi possano concordare la chiave di persona. Questa seconda ipotesi è chiaramente molto costosa da un punto di vista dell'efficienza del sistema, ma come vedremo è in qualche caso l'unica praticabile.

Un'analogia molto usata per raffigurare il processo di trasmissione sicura dei dati riservati è quella di pensare a una cassetta blindata chiusa con un lucchetto e contenente il documento riservato. La cassa chiusa viene fatta transitare sul canale, ed entrambi gli utenti legittimi X e Y devono possedere la chiave del lucchetto. Se è X ad acquistare il lucchetto, egli dovrà poi mandare una copia della chiave a Y, usando un corriere fidato e non corruttibile.

Gestione delle chiavi

Anche il trasferimento a distanza della chiave non è però operazione esente da problemi, giacché l'eventuale intercettazione della stessa da parte della spia,

sempre appostata sul canale, comprometterebbe la sicurezza dell'intero sistema. Non si può , d'altra parte, sperare di cifrare la chiave, poiché si dovrebbe comunque concordare una chiave del second'ordine per questa seconda cifratura ... . L'unica scappatoia è allora quella di ritornare, con la sola chiave, sul piano del mascheramento sintattico, *impedendo* l'accesso al canale sul quale transita la chiave. Ciò costituisce una contraddizione per la natura stessa della crittologia, poiché disponendo di un canale inaccessibile e sicuro vi si potrebbe far confluire tutta l'informazioni riservata che costituisce il messaggio. Questa contraddizione viene però sanata dal momento che l'informazione legata alla chiave è di solito quantitativamente molto inferiore all'informazione generalmente associata al messaggio, cosicché con una singola chiave di dimensione modesta si può smaltire una quantità ingente d'informazione. Ciò rende allora verosimile l'impiego di un canale speciale, fidato e inaccessibile, per inoltrare a destinazione la chiave, anche se esso dovesse essere molto costoso o disponibile solo per brevi periodi.

Si osservi che la condivisione da parte dei due utenti legittimi della segretezza della chiave risolve in una qualche misura anche il problema degli attacchi attivi da parte dell'oppositore, poiché l'acquisizione da parte dell'utente legittimo di un crittogramma decifrabile fornisce una garanzia indiretta anche sull'autenticità della sorgente, la sola che conosce la chiave k e quindi, in linea di principio, la sola che può aver prodotto un crittogramma che porta a una decifrazione significativa. Ribadiamo però che ciò non vale in senso stretto, poiché un oppositore potrebbe scegliere un crittogramma a caso nella speranza che la decifrazione con la chiave legittima porti a un messaggio significativo (si veda a tal proposito il paragrafo 10.3.3). Parimenti potrebbe modificare il crittogramma in transito in modo casuale, sempre sperando che la decifrazione sia coerente. Anche se la modificazione non comporta il *controllo* da parte del frodatore delle informazioni inserite (o rimosse) fraudolentemente, il successo di un simile attacco è in ogni caso da evitare.

Un approccio alternativo per evitare lo scambio della chiave tra due utenti legittimi X e Y può essere messo in luce ricorrendo alla seguente analogia. Si supponga di dotare i due utenti di due chiavi segrete e personali x e y. Supponiamo che X voglia spedire a Y un messaggio segreto m, che viene riposto in una cassetta blindata che può essere chiusa con dei lucchetti. X chiude la cassetta con il proprio lucchetto C_x, e spedisce a destinazione la cassetta. Sul canale nessuno dispone della chiave D_x per aprire il lucchetto C_x, e la segretezza non viene compromessa. Quando la cassetta arriva a destinazione Y vi aggiunge anche il proprio lucchetto C_y, rispedendo il tutto nuovamente a X. A questo punto X può aprire il proprio lucchetto C_x impiegando la chiave D_x, e rispedire sul canale la cassetta a Y, che avrà finalmente la possibilità di leggere il messaggio

aprendo il lucchetto C_y mediante la propria chiave D_y. Il protocollo di scambi è riassunto nel modo seguente

$$\begin{array}{lll} X, x: & C_x(m) & \longrightarrow Y \\ Y, y: & C_y(C_x(m)) & \longrightarrow X \\ X, x: & D_x(C_y(C_x(m))) = C_y(m) & \longrightarrow Y \\ Y, y: & D_y(C_y(m)) = m & \end{array} \tag{7.2}$$

nel quale è sottintesa l'ipotesi di *commutatività* delle operazioni di cifratura/-decifrazione (l'ordine di applicazione delle funzioni è irrilevante). In tutte le transazioni sul canale il sistema è protetto da almeno un lucchetto e i due utenti non necessitano di scambiarsi le chiavi, poiché ciascuno usa solo la propria. Il grosso svantaggio di questo sistema è che l'utente Y potrebbe facilmente essere impersonato da un utente non autorizzato F, e X non sarebbe in grado di accorgersene consegnando il messaggio segreto nelle mani del frodatore.

Una terza possibilità per lo scambio della chiave segreta, messa in luce solo recentemente con l'introduzione delle funzioni unidirezionali e in generale con l'ingresso nella nuova era della crittografia a *chiave pubblica*, è quella di fare in modo che X e Y possano generare per loro conto una chiave $k(X, Y)$ senza il bisogna d'incontrarsi fisicamente, avvalendosi contemporaneamente di un'informazione segreta, ma personale e di cui ciascuno è depositario, e di un informazione pubblica, relativa all'utente con cui si intende scambiare i messaggi riservati. L'analogia dei lucchetti subisce la seguente trasformazione, da lucchetti per così dire "in parallelo" a lucchetti "concatenati".

Supponiamo che i due utenti si scambino i propri lucchetti aperti C_x e C_y (oppure che esista un "ufficio postale" dove i lucchetti sono depositati aperti e disponibili al pubblico). Se X vuole mandare a Y un messaggio riservato usando un lucchetto comune C_{xy}, apribile da entrambi, egli chiude il proprio lucchetto sul lucchetto di Y e sul coperchio della cassa, per poi chiudere il lucchetto di Y sulla parte fissa della cassa, creando così il lucchetto concatenato C_{xy}. In questo modo tanto X che Y possono riaprire la scatola, che rimane chiusa da una chiave "condivisa".

Queste considerazioni sulle chiavi introducono al problema generale di una loro gestione, o *distribuzione*, soprattutto nei contesti in cui ci sia una rete con molti utenti che devono comunicarsi l'un l'altro delle informazioni riservate; tale riservatezza non va infatti garantita solo rispetto agli utenti esterni alla rete, che sono non autorizzati per definizione, ma anche nei confronti degli altri utenti autorizzati della rete, che non sono però legittimati a intercettare e leggere messaggi che non siano loro indirizzati.

L'efficacia della protezione associata allo schema (7.1) risiede in generale

nel fatto che lo spazio delle chiavi, pur essendo molto più piccolo dello spazio dei messaggi, ha comunque una cardinalità tale da scoraggiare una *ricerca esauriente* . Naturalmente questa è una condizione necessaria per la sicurezza, ma non certo sufficiente, giacché in alcuni casi il cifrario nasconde delle debolezze strutturali, o legate ad una scelta poco felice della chiave, che possono consentire attacchi molto più sofisticati di quello basato su una semplice e rozza ricerca esauriente.

In altre parole la *decrittazione*, cioè il tentativo da parte della spia di recuperare l'informazione riservata avendo a disposizione il solo crittogramma, è in genere un'operazione piuttosto raffinata, che tende a sfruttare tutti i punti deboli del sistema, tanto quelli noti al progettista, quanto quelli di cui fosse malauguratamente inconsapevole.

Chiave segreta e chiave pubblica

Chiudiamo questo quadro generale riferendo sulla nuova fisionomia che la crittologia ha assunto a partire dagli anni '70, più precisamente dalla pubblicazione nel 1976 del celebre articolo di Diffie e Hellman "New Directions in Cryptography" [25]. Fino a quel momento i sistemi crittografici, anche se basati su cifrari diversi, avevano comunque in comune il fatto che la chiave dovesse essere mantenuta strettamente segreta; anzi, questo fatto è sempre stato considerato una sorta di postulato, senza il quale non si può neanche parlare di crittografia. A ben osservare si scopre però che in realtà le chiavi sono due, una *diretta o di cifratura* che porta dal messaggio al crittogramma, e l'altra *inversa o di decifrazione* che restituisce il messaggio a partire dal crittogramma. Il fatto che in pratica si faccia riferimento a una sola chiave dipende dalla circostanza che data la chiave di cifratura si ottiene in modo pressochè immediato anche quella di decifrazione, *cioè trasformazione diretta e trasformazione inversa sono facilmente riconducibili l'una all'altra*.

Pensando all'analogia dei lucchetti si è insomma sempre implicitamente supposto che l'unico modo per consentire l'apertura a Y di una cassetta chiusa da X fosse quello di spedire a distanza una copia della chiave originale, dotando entrambi gli utenti *essenzialmente* della stessa chiave. L'analogia dei lucchetti, nella sua semplicità, consente però di farci comprendere che cifratura C_k e decifrazione D_k possono essere operazioni strutturalmente diverse e fortemente asimmetriche; da un punto di vista matematico si possono cioè costruire delle coppie di funzioni C_k e D_k che presentano una forte asimmetria, nel senso che risulta molto difficile ricavare una funzione dall'altra o, se vogliamo, *invertire* la cifratura C_k in modo da ottenere la decifrazione D_k. In altre parole si possono escogitare dei cifrari in cui la conoscenza di una delle due chiavi, per esempio quella di cifratura, non implica assolutamente la conoscenza immediata

della chiave inversa; anzi si può fare in modo che lo sforzo computazionale per ricavare la chiave inversa dalla diretta sia praticamente insostenibile.

Questo nuovo punto di vista offre vantaggi straordinari, che si innervano nello schema del cifrario consentendo addirittura la *pubblicazione* della funzione di cifratura C_y di un certo utente Y, poichè la sua conoscenza, anche da parte della spia, non consente di ricavare la chiave inversa di decifrazione $D_y = C_y^{-1}$, che rimane segreta e nota solo a Y. Se dunque X vuole spedire un messaggio m riservato a Y, egli deve consultare un elenco pubblico delle chiavi di cifratura, curato p.es. dal gestore della rete, leggere la C_y e applicare la trasformazione

$$X:\ c = C_y(m) \qquad \stackrel{canale}{\longrightarrow} \qquad Y:\ m = D_y(c) = D_y C_y(m) \tag{7.3}$$

In ricezione l'utente legittimo Y, che conosce (ed è il solo) la chiave inversa D_y, la applica al crittogramma ricevuto riottenendo il messaggio. Tutto il procedimento si basa evidentemente sul fatto che il problema di ricavare la chiave inversa da quella diretta, che chiamiamo per comodità *problema inverso*, è computazionalmente *intrattabile*.

Oss. 7.1. Si osservi che la trasmissione ha successo dal punto di vista della sicurezza solo se diamo per scontato che l'elenco pubblico non sia falsificabile; altrimenti un falsario F potrebbe cambiare il valore della chiave pubblica di Y (mettendo p.es. la propria) e decifrare sistematicamente tutti i messaggi destinati a Y. Se F usa inoltre la precauzione di cifrarli nuovamente con la vera chiave C_y di Y, né X né Y si accorgeranno di nulla. Ciò apre il problema della *certificazione delle chiavi*, anch'esso parte integrante della moderna crittologia.

Accanto alla crittografia a chiave segreta o *simmetrica*, esiste dunque una crittografia a chiave pubblica, o *asimmetrica*, i cui sviluppi sono legati da una parte all'esistenza di trasformazioni con le caratteristiche sopra specificate, che chiamiamo *funzioni unidirezionali*, dall'altra alla *teoria della complessità*, che deve fornire gli elementi teorici necessari ad attestare l'intrattabilità del problema inverso, e dunque la sicurezza del sistema.

Ritornando all'analogia dei lucchetti si pensi al seguente semplice schema di cifratura asimmetrica : quando X deve spedire un messaggio a Y quest'ultimo trasmette a X il proprio lucchetto aperto C_y; X lo usa per chiudere la cassa, spedendo il tutto a Y, che è l'unico in grado di riaprire la cassa.

Il tutto si basa sulla constatazione a dir poco banale che *un lucchetto può essere chiuso da chiunque, ma riaperto solo dal legittimo proprietario*! Sembra a questo punto incredibile che, a fronte di una gloriosa storia che parte dai geroglifici egiziani e che ha anche nella Bibbia degli esempi di secretazione delle informazioni, la crittografia sia riuscita a materializzare un sistema di cifratura a *chiave asimmetrica* solamente a metà degli anni '70, anche se la giustificazione di ciò è data dal fatto che la teoria della complessità è una disciplina piuttosto recente. In tale sistema si consente la pubblicazione della chiave di cifratura e ci si affranca dall'annoso problema della distribuzione delle chiavi, che per sistemi con molti utenti diventa di difficilissima attuazione.

Ipotesi di lavoro

Per quanto attiene all'atteggiamento da usare nei confronti del crittanalista, è opportuno porsi sempre nelle condizioni più sfavorevoli, immaginando che egli disponga di ingenti risorse computazionali, di un tempo grande a piacimento per eseguire i calcoli (anche se in pratica limitato), e che sia sempre in grado d'intercettare qualunque crittogramma in transito sul canale (o depositato in memoria).

A queste considerazioni di carattere generale bisogna però aggiungere altri elementi specifici, connaturati al contesto fisico nel quale si svolge l'intercettazione dei crittogrammi.

All'inizio si è detto che la trasformazione che costituisce la chiave è l'informazione segreta che consente di recuperare il messaggio. Tuttavia si potrebbe tentare di tenere segreta anche la *struttura* del cifrario usato, cioè la classe cui appartiene la famiglia di trasformazioni. Ciò consente di avere un vantaggio ulteriore sul crittanalista, poiché quest'ultimo, prima di iniziare a cercare la chiave, sarà impegnato a individuare il cifrario che è stato usato. Nel passato tale tecnica è stata usata frequentemente, soprattutto negli ambienti militari dove la crittografia ha sempre avuto un ruolo rilevante. Oggi un tale vantaggio risulterebbe illusorio, al punto che la pubblicità sul cifrario usato costituisce, se il cifrario è considerato sicuro, una sorta di deterrente nei confronti della spia. Ci si affida allora al cosiddetto *principio di Kerckhoffs*, che stabilisce che *tutta la sicurezza del cifrario dev'essere affidata alla chiave*, giacché si presume che il crittanalista conosca nei dettagli la struttura del cifrario usato.

Naturalmente queste considerazioni non hanno un carattere assoluto; come celebre controesempio si pensi al *manoscritto di Voynich*, documento cifrato di ben 204 pagine la cui scoperta risale al 1666 (la datazione della stesura del documento non è però nota con precisione), che ha resistito finora a ben 334 anni di massicci attacchi crittanalitici, attuatiti con le tecniche più raffinate, e che hanno impegnato crittanalisti di chiara fama (tra cui Friedman, cui si deve un metodo di crittanalisi per cifrari polialfabetici; vedi paragrafo 7.3.2).

Un altro elemento da tenere in considerazione è il seguente. La macchina cifrante, qualunque essa sia, potrebbe essere accessibile alla spia, anche se solo per poco tempo. Si pensi al caso pratico di un *Centro di Comando* militare e all'ipotesi che la *Sezione Cifra* cada in mano al nemico. In tal caso la spia ha l'opportunità d'inserire un qualunque messaggio nella macchina e di rilevare il crittogramma corrispondente. Un tale contesto è noto col nome di *attacco crittanalitico con testo in chiaro* ed è particolarmente pericoloso per l'incolumità dell'intero cifrario. Ci si deve dunque cautelare anche da questa ipotesi, verificando che la conoscenza di una porzione c di crittogramma e del corrispondente messaggio m non comprometta la segretezza della chiave, che potrebbe essere ricavata in linea di principio dal confronto tra m e c.

Ricapitolando, si dovranno fare le seguenti

Ipotesi di lavoro. Il crittanalista dispone:

1. della completa conoscenza del cifrario usato;
2. di una quantità potenzialmente illimitata di crittogramma;
3. di una porzione di crittogramma e del corrispondente messaggio (eventualmente scelto dallo stesso crittanalista).

7.1.2 Autenticazione

Segretezza e autenticità di un documento sono stati storicamente legati dal fatto che si è fatto sempre riferimento a sistemi di cifratura a chiave simmetrica (o chiave segreta), nei quali la condivisione della segretezza della chiave fornisce una qualche forma di garanzia indiretta anche sull'autenticità del messaggio o del mittente (anche se tale garanzia non è assoluta; si veda a tal riguardo il paragrafo 10.3.3). Solo recentemente, con l'avvento della crittografia a chiave pubblica, si è realizzato che si tratta di esigenze distinte per loro natura, e che necessitano soluzioni differenziate e indipendenti: si può aver infatti segretezza senza autenticazione e viceversa.

Il problema è stato affrontato nei suoi caratteri generali da G. Simmons [81], che analogamente a quanto fatto da Shannon per la segretezza è riuscito a porre le basi per una *teoria generale dell'autenticazione*, di cui daremo cenno nella sezione 10.3.3.

Per il momento si osservi che l'approccio basato sulla chiave asimmetrica consente di separare in modo naturale le due esigenze di segretezza e autenticazione. Ciò deriva dal fatto che chiunque può usare la chiave pubblica C_y per trasmettere un messaggio segreto a Y. Per di più un'eventuale manipolazione delle chiavi pubbliche di cifratura, come sottolineato nell'osservazione 7.1, non fornisce garanzie a Y neanche nel caso in cui X abbia annunciato un suo imminente messaggio.

Riprendiamo in considerazione la funzione di cifratura C_x e la funzione di decifrazione D_x, sempre supponendo che C_x sia unidirezionale e che la sua conoscenza non consenta di ricavare D_x. L'autenticazione si fa nel modo seguente: poiché l'utente X, che dev'essere autenticato, è l'unico che conosce la trasformazione D_x, egli la applica al proprio nome X ottenendo $D_x(X)$, che viene inviata sul canale. Dall'altra parte Y applicherà la chiave pubblica C_x a quanto ricevuto, ottenendo la certificazione di X.

$$X:\ f_x = D_x(X) \qquad \stackrel{canale}{\longrightarrow} \qquad Y : X = C_x(f_x) = C_x D_x(X) \tag{7.4}$$

Dunque, chiunque può controllare l'identità di X e solo X è in grado di generare la propria *firma* $f_x = D_x(X)$. Si faccia però attenzione al fatto che tale firma potrebbe essere reimpiegata da un falsificatore in un momento successivo; di conseguenza bisogna associare all'identità di X alcune informazioni di carattere temporale, o contestuale, che possano far emergere l'eventuale truffa.

Nel caso in cui sia necessario autenticare un messaggio riservato m che X vuole spedire a Y, coniugando segretezza e autenticazione in un'unica trasmissione, la procedura è data dalla composizione delle (7.3) e (7.4), secondo il seguente schema: X applica a m la propria chiave privata D_x, ottenendo $f_x = D_x(m)$; a questo messaggio "firmato" applica la chiave pubblica C_y di Y, ottenendo il crittogramma $c = C_y D_x(m)$, che viene spedito sul canale. L'unico utente in grado di decifrare c è Y, che possiede la chiave di decifrazione D_y mediante la quale estrae il messaggio firmato di X, che può essere autenticato applicando la funzione di cifratura pubblica C_x che Y legge dall'elenco ufficiale degli utenti della rete. I passaggi sono i seguenti:

$$\begin{aligned} X : m \xrightarrow{\text{firma}} f_x = D_x(m) \xrightarrow{\text{cifratura}} \quad & c = C_y D_x(m) \xrightarrow{\text{canale}} \\ \xrightarrow{\text{canale}} \quad Y : c = C_y D_x(m) \xrightarrow{\text{decifrazione}} \quad & f_x = D_y C_y D_x(m) = D_x(m) \\ \xrightarrow{\text{autenticazione}} \quad & m = C_x(f_x) = C_x D_x(m) \end{aligned} \tag{7.5}$$

7.2 Cifrari puri e classi residue

Affrontiamo ora l'impostazione che fu data da Shannon, nel suo celebre articolo "Communication Theory of Secrecy Systems" [78], all'impianto teorico della crittografia. Riprendiamo la definizione 7.1 e analizziamo meglio la struttura del grafo associato al cifrario. In generale il problema è quello di riuscire a capire su quali parametri bisogna agire per aumentare la sicurezza dello stesso cifrario. Una prima proprietà, molto importante sul piano operativo, è la definizione di *purezza di un cifrario*; essa consente di introdurre le *classi residue* dei messaggi e dei crittogrammi, che forniscono indicazioni relative alla sicurezza.

Siano dunque assegnati lo spazio dei messaggi $\mathcal{M}$, dei crittogrammi $\mathcal{C}$ e delle chiavi $\mathcal{K}$. Siano inoltre $T_i, T_j, T_k, \ldots$ chiavi diverse appartenenti a $\mathcal{K}$, con $T_i^{-1}, T_j^{-1}, T_k^{-1}, \ldots$ le rispettive trasformazioni inverse.

Def. 7.2. *Un cifrario si dice puro se, assegnata una qualunque terna T_i, T_j, T_k di $\mathcal{K}$, esiste una trasformazione $T_h \in \mathcal{K}$ tale che*

$$T_h = T_i T_j^{-1} T_k \tag{7.6}$$

La purezza di un cifrario garantisce che la cifratura di un messaggio m con una chiave qualsiasi T_k, seguita da una decifrazione T_j^{-1} e da una nuova cifratura T_i (che porta al crittogramma $c = T_i T_j^{-1} T_k(m)$) si può ottenere mediante una sola cifratura T_h, che trasforma direttamente da m in c (si veda la figura 7.2). Prendiamo ora un messaggio m, cifrandolo e decifrandolo usando tutte le chiavi

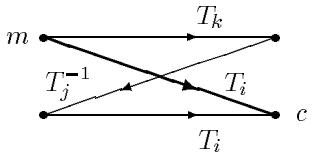


Figura 7.2 Purezza del cifrario.

possibili. L'insieme dei messaggi $M(m) \in \mathcal{M}$ che così si ottiene si chiama *classe residua dei messaggi*. Se il cifrario è puro accadono i seguenti fatti:

- la classe residua $M(m^*)$ di ogni messaggio di $m^* \in M(m)$ è la $M(m)$ stessa, che è una classe di equivalenza. Si prenda infatti $\widehat{m} \in M(m^*)$, cioè $\widehat{m} = T_p^{-1} T_i(m^*)$ per qualche i e p, con $m^* = T_j^{-1} T_k(m)$ per qualche j e k, in quanto $m^* \in M(m)$. Si ha

$$\widehat{m} = T_p^{-1} T_i(m^*) = T_p^{-1} T_i T_j^{-1} T_k(m) = T_p^{-1} T_h(m)$$

  dove l'ultima uguaglianza deriva dall'ipotesi di purezza. La relazione di equivalenza che rimane definita è allora

$$m_1 \sim m_2 \text{ se } \exists\, T_i T_j : m_1 = T_j^{-1} T_i(m_2)$$

- assegnata la classe $M(m)$ definiamo come $C(m)$ l'insieme di tutti i crittogrammi ottenibili cifrando m in tutti i modi possibili. $C(m)$ è la *classe residua dei crittogrammi* ed è la stessa per tutti i messaggi $\widehat{m} \in M(m)$. Essa è l'immagine della classe residua dei messaggi $M(m)$. Anche in questo caso l'ipotesi di purezza gioca un ruolo fondamentale. Se $\widehat{m} \in M(m)$ sia $C(\widehat{m})$ la classe residua dei crittogrammi generata da $\widehat{m}$. Se $\widehat{c} = T_i(\widehat{m})$ è un crittogramma di $C(\widehat{m})$, tenendo conto che $\widehat{m} = T_j^{-1} T_k(m)$ in quanto $\widehat{m} \in M(m)$ si ha

$$\widehat{c} = T_i(\widehat{m}) = T_i T_j^{-1} T_k(m) = T_h(m)$$

  sempre per l'ipotesi di cifrario puro, e dunque $\widehat{c} \in C(m)$.

Se ci poniamo ora dal punto di vista del crittanalista che abbia intercettato il crittogramma c della classe di $C(m)$ sappiamo che $m \in M(m)$. Tuttavia, scegliendo la chiave appropriata ogni messaggio di $M(m)$ fornisce lo stesso crittogramma c. Se inoltre i messaggi sono equiprobabili non avremo elementi per decidere quale chiave sia stata usata. $M(m)$ può allora essere considerato come l'insieme dei messaggi che il crittanalista può "confondere" tentando una decrittazione con tutte le chiavi possibili a partire da c. Poiché $C(m)$ contiene crittogrammi ai quali si arriva scegliendo tutte le chiavi possibili per gli elementi di $M(m)$, avremmo potuto ottenere gli stessi risultati a partire da un qualunque altro crittogramma c^* di $C(m)$. I crittogrammi di $C(m)$ sono dunque *crittanaliticamente equivalenti* dal punto di vista dell'intercettatore.

Per frustrare le possibilità di successo del crittanalista è allora necessario che la classe $M(m)$ sia molto numerosa, in modo da cautelarsi quantomeno rispetto alla ricerca esauriente.

La struttura dello spazio dei messaggi di un cifrario puro viene specificata dal punto di vista algebrico dal seguente

Teorema 7.1. *In un cifrario puro $\boldsymbol{T}$ le operazioni $T_j^{-1}T_i$ che trasformano lo spazio dei messaggi in sé formano un gruppo $G(\boldsymbol{T})$ di permutazioni il cui ordine è pari al numero delle chiavi.*

Dim. Innanzitutto bisogna verificare che l'insieme sia chiuso, cioè che presi due elementi di $G(\boldsymbol{T})$ la loro composizione appartenga ancora a $G(\boldsymbol{T})$. A tal riguardo si ha

$$T_p^{-1}T_i \cdot T_j^{-1}T_k = T_p^{-1}T_h \in G(\boldsymbol{T})$$

poiché per la purezza $\exists T_h : T_h = T_i \cdot T_j^{-1}T_k$. La verifica che si tratta di un gruppo è immediata, poiché la composizione è associativa, esiste l'elemento neutro $(T_j^{-1}T_j)$ e inoltre $T_j^{-1}T_i$ ha come inverso $(T_j^{-1}T_i)^{-1} = T_i^{-1}T_j$, che è sempre un elemento di $G(\boldsymbol{T})$. Per quanto riguarda l'ordine, se $T_1, T_2, \ldots, T_m$ sono le m chiavi, prendendo un elemento $T_i^{-1}T_j$ e tenendo fisso i si ottengono m elementi distinti, e dunque $|G(\boldsymbol{T})| \geq m$. D'altra parte, scegliendo in modo indipendente i e j tra le m possibilità si ha $|G(\boldsymbol{T})| \leq m^2$. Si tenga però conto che ogni elemento del tipo $T_j^{-1}T_k$ può in realtà essere espresso in m modi distinti; infatti per qualunque terna T_i, T_j, T_k si ha $T_h = T_iT_j^{-1}T_k$ per la purezza del cifrario e dunque, tenendo fisso $T_j^{-1}T_k$ e scegliendo T_i in m modi diversi si ottengono m elementi del tipo $T_i^{-1}T_h = T_j^{-1}T_k$. Di conseguenza bisogna dividere per m ottenendo $|G(\boldsymbol{T})| \leq m^2/m = m$. Per confronto con la $|G(\boldsymbol{T})| \geq m$ si ottiene la tesi □

Il gruppo $G(\boldsymbol{T})$ ripartisce lo spazio dei messaggi nelle classi di equivalenza prima definite come classi residue dei messaggi, e tale ripartizione ne induce una

analoga nello spazio dei crittogrammi (le classi residue dei crittogrammi). Quest'ultima è determinata dall'esistenza del gruppo $G'(\boldsymbol{T})$ delle trasformazioni tra crittogrammi del tipo $T_i T_j^{-1}$. Per ogni T_i vale la relazione

$$G'(\boldsymbol{T}) = T_i \, G(\boldsymbol{T}) \, T_i^{-1}$$

i due gruppi sono coniugati e hanno lo stesso ordine m.

Nella figura 7.3 si può vedere un esempio di cifrario puro e delle corrispondenti classi residue.

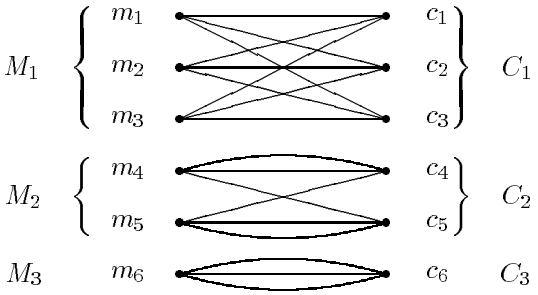


Figura 7.3 Classi residue dei messaggi e dei crittogrammi in un cifrario puro.

7.3 Alcuni esempi di cifrari classici

Prima di continuare l'analisi teorica impostata da Shannon forniamo alcuni esempi di cifrari, considerati oggi di interesse meramente storico-didattico se presi individualmente, ma che sono però la base di sistemi composti di cifratura ancora attuali e apprezzati (si veda per esempio il sistema DES della sezione 8.1.1). In particolare analizzeremo le due famiglie di cifrari a *sostituzione* (semplice e polialfabetica) e a *trasposizione*. A queste due categorie di base, e alle possibili combinazioni tra le due, si possono ricondurre quasi tutti i cifrari storici che si sono succeduti nel corso dei secoli, e in generale tutti i *cifrari a blocco*, nei quali a una porzione di messaggio di n lettere su un alfabeto K-ario viene fatta corrispondere una porzione di n lettere di cifrario basata sullo stesso alfabeto.

I cifrari classici sono utili anche per impratichirsi con la distanza, spesso abissale, che separa un attacco crittanalitico serio dalla rozza ricerca esauriente della chiave. Come vedremo, infatti, anche a fronte di cardinalità rilevanti per lo spazio delle chiavi l'ingegno del crittanalista può mettere in luce carenze strutturali che consentono di forzare il sistema in poche battute.

In questi primi esempi immaginiamo di operare con messaggi scritti in un linguaggio naturale, p.es. l'Italiano, col corrispondente alfabeto $\mathcal{A}$ di 21 lettere.

7.3.1 Cifrario a sostituzione semplice

Assieme alla trasposizione è la più immediata delle trasformazioni, e consiste in una permutazione delle lettere dell'alfabeto

$$A, B, C, \ldots, Z \quad \xrightarrow{Cif} \quad \pi(A), \pi(B), \pi(C), \ldots, \pi(Z)$$

Le possibili chiavi sono $21!$, compresa la chiave degenere che consiste nell'associare ciascuna lettera a sé stessa. L'esempio mette in luce il significato della definizione 7.1, nella quale la famiglia delle trasformazioni è in questo caso l'insieme completo di tutte le permutazioni, mentre la scelta della chiave corrisponde all'impiego di una fra le possibili $21!$ permutazioni. Le ipotesi di lavoro viste nella sezione 7.1.1 prevedono che il crittanalista sappia che il messaggio è stato cifrato usando il cifrario a sostituzione semplice, ma non conosca quale, fra le $21!$ chiavi, sia stata effettivamente usata. Tutta la sicurezza del cifrario dev'essere dunque affidata alla chiave.

La decifrazione avviene usando la chiave inversa

$$\pi(A), \pi(B), \pi(C), \ldots, \pi(Z) \quad \xrightarrow{Dec} \quad A, B, C, \ldots, Z$$

che si ricava immediatamente da quella diretta (basta leggere la tabella delle corrispondenze nel senso inverso); il sistema è dunque a chiave segreta o simmetrica. Un sistema di cifratura a chiave pubblica si differenzierebbe proprio su questo punto, poiché una delle due chiavi non dovrebbe essere ottenibile dall'altra se non con uno sforzo computazionale superiore alle risorse del crittanalista.

Sul principio della sostituzione sono basati numerosi cifrari d'interesse storico, come il *cifario di Cesare*, che consiste nell'associare a ciascuna lettera quella che nell'alfabeto la segue di 3 posizioni, cioè $\pi(A) = D, \pi(B) = E, \pi(C) = F, \ldots$ (prolungando l'alfabeto per circolarità in modo che la A sia seguita dalla Z), o la sua generalizzazione che porta al *cifrario a rotazione*, in cui al posto della traslazione di ordine 3 si usa una traslazione j, con $0 \leq j \leq 20$. Si noti che questi due cifrari sono in realtà sottofamiglie della famiglia completa di tutte le permutazioni; il cifrario a rotazione ha infatti solamente 21 chiavi, mentre il cifrario di Cesare è addirittura *degenere*, in quanto ha un'unica chiave.

Nel caso della sostituzione semplice si dà per scontato che la stessa chiave venga mantenuta nel corso dell'intero messaggio. Anche se il numero di chiavi possibili è tutto sommato rilevante ($21! \approx 5.11\ 10^{19}$) la forzatura del cifrario è immediata, perlomeno nel caso di un linguaggio naturale. Si possono infatti usare le informazioni statistiche sulle frequenze relative delle singole lettere, dei digrammi, trigrammi ecc. che si possono ricavare da frasi di senso compiuto scritte nel linguaggio d'interesse. Nella tabella 7.1 sono riportate come esempio le frequenze individuate nel romanzo "I Promessi Sposi", depurato naturalmente degli spazi e della punteggiatura, che costituiscono sicuramente un'ottima approssimazione per la lingua italiana. Il testo si compone infatti di oltre un milione di caratteri

E	0.12059	L	0.05567	V	0.02305
A	0.11512	S	0.05459	G	0.01713
O	0.09640	C	0.04692	H	0.01352
I	0.09530	D	0.03731	F	0.01052
N	0.07292	U	0.03569	B	0.00973
R	0.06599	P	0.02967	Q	0.00779
T	0.06083	M	0.02365	Z	0.00760

Tabella 7.1 Tabella delle frequenze relative delle lettere in Italiano (ricavate da "I Promessi Sposi").

Esempio 7.1. Prendiamo il caso di un cifrario a sostituzione completo, che faccia cioè uso di tutte le possibili chiavi, e che realizzi la seguente permutazione

A B C D E F G H I L M N O P Q R S T U V Z
$G\ V\ R\ C\ Q\ U\ D\ M\ A\ O\ T\ L\ N\ B\ F\ Z\ P\ H\ S\ I\ E$

Se la parte iniziale del messaggio intercettato è

$$BQZTQPAIGLQOOGRAHHGCNOQLHQ\ldots$$

(gli spazi, cioè le *nulle*, vengono eliminati) il primo passo della decrittazione procede calcolando la frequenza relative delle lettere del crittogramma; si supponga di ricavare la seguente tabella, ordinata per valori decrescenti

G	Q	A	N	H	...	U	E	F
0.120	0.115	0.100	0.095	0.075	...	0.009	0.006	0.004

Essendo G la lettera più frequente, sulla base della tabella (7.1) è verosimile che essa corrisponda a una E, mentre si può supporre in prima battuta che la Q corrisponda a una A. Come prima iterazione si può dunque tentare una sostituzione delle lettere basata sul confronto tra gli ordinamenti delle due tabelle. Il risultato che si ottiene potrebbe essere

PARMASOVERALLECONNEDILARNA ...

che non è ancora leggibile. Tuttavia lo scambio di alcune lettere che hanno frequenze relative molto vicine porta subito alla soluzione. Per esempio scambiando la A con la E si ottiene "PERMESOVARELLACONNADILERNE ... " che costituisce un buon punto di partenza. Con un altro paio di tentativi (p. es. lo scambio di O con I ecc.) si giunge alla decrittazione completa, che riappropiandosi delle nulle

PER ME SI VA NELLA CITTA DOLENTE ...

È questo un esempio di *crittanalisi statistica* di un cifrario a sostituzione.

Anche se l'esempio precedente è un po' semplificato per fini didattici, la decrittazione non presenta comunque particolari dificoltà, soprattutto se si hanno a disposizione anche le tabelle con le frequenze relative del secondo o terzo ordine.

7.3.2 Cifrario a sostituzione polialfabetica

Il punto debole del cifrario a sostituzione deriva dalla struttura statistica del linguaggio usato, e in particolare dalla disomogeneità delle frequenze relative delle lettere singole, dei digrammi, trigrammi, che induce una simile disomogeneità nel crittogramma. Un metodo per cautelarsi potrebbe essere quello di ottenere, mediante opportune trasformazioni, una distribuzione (più) omogenea delle frequenze relative. A tale scopo si può ricorrere a una *espansione* dei dati, usando come alfabeto per il crittogramma un'estensione n-esima dell'alfabeto di partenza (cifrario *omofonico*, non inquadrabile nella definizione 7.1). Per $n = 2$ abbiamo dunque 21^2 digrammi da associare alle 21 lettere dell'alfabeto in chiaro; di conseguenza possiamo usare *più* coppie diverse per cifrare la stessa lettera in chiaro, soprattutto quando quest'ultima sia caratterizzata da una frequenza relativa elevata; ciò consente di ottenere un appiattimento delle frequenze relative di primo ordine sul crittogramma. L'aumento della difficoltà nella crittanalisi non è però tale da giustificare un'espansione così massiva dei dati; se il crittogramma è sufficientemente lungo il ricorso alle tecniche statistiche tradizionali, sia pure relative agli ordini statistici superiori al primo, consente comunque una decrittazione efficace del cifrario omofonico.

Un'ovvia generalizzazione del cifrario a sostituzione semplice è quella di usare non una, ma *diverse* chiavi distinte all'interno dello stesso messaggio. In altre parole, se $m_1 m_2 \dots m_h$ ($m_i \in \mathcal{A}$) è un blocco-messaggio di lunghezza h, per la sua cifratura ricorreremo alle permutazioni $\pi_{i_1}, \pi_{i_2}, \dots, \pi_{i_h}$, con $1 \leq i_j \leq 21!$

$$m_1, m_2 \dots m_h \qquad \xrightarrow{Cif} \qquad \pi_{i_1}(m_1), \pi_{i_2}(m_2), \dots, \pi_{i_h}(m_h)$$

Questa tecnica è denominata *cifratura polialfabetica*. Un celebre esempio di cifrario polialfabetico, che generalizza i cifrari a rotazione, è dato dal *cifrario di Vigenère*. Storicamente esso venne attuato scegliendo una parola chiave, p.es. "MORMOLICE", e ripetendo le lettere della parola in corrispondenza delle lettere del messaggio

$$\begin{array}{l} \text{P ERMES I VANEL L AC I TT A DO L ENTE} \dots \\ \text{MORMOL I CEMORMO L I CEMORMOL I C} \dots \\ \\ CSLZSESAEASDVONSVAMRGVSZEG \dots \end{array} \tag{7.7}$$

A questo punto ciascuna lettera della parola chiave determina la codifica della lettera A del messaggio, indicando il valore della corrispondente traslazione. Nell'esempio la M iniziale di $MORMOLICE$ ci indica che la "A" diventa "M", la "B" diventa "N" e così via, individuando il valore $j = 10$ che porta la

"P" in una "C". La seconda lettera è una "O", che implica $A \to O$, $B \to P$, $C \to Q$, ecc.

Le corrispondenze si possono leggere in modo immediato dalla tabella di Vigenère sotto riportata

A B C D E F G H I L M N O P Q R S T U V Z
B C D E F G H I L M N O P Q R S T U V Z A
C D E F G H I L M N O P Q R S T U V Z A B
D E F G H I L M N O P Q R S T U V Z A B C
E F G H I L M N O P Q R S T U V Z A B C D
F G H I L M N O P Q R S T U V Z A B C D E
G H I L M N O P Q R S T U V Z A B C D E F
H I L M N O P Q R S T U V Z A B C D E F G
I L M N O P Q R S T U V Z A B C D E F G H
L M N O P Q R S T U V Z A B C D E F G H I
M N O P Q R S T U V Z A B C D E F G H I L
N O P Q R S T U V Z A B C D E F G H I L M
O P Q R S T U V Z A B C D E F G H I L M N
P Q R S T U V Z A B C D E F G H I L M N O
Q R S T U V Z A B C D E F G H I L M N O P
R S T U V Z A B C D E F G H I L M N O P Q
S T U V Z A B C D E F G H I L M N O P Q R
T U V Z A B C D E F G H I L M N O P Q R S
U V Z A B C D E F G H I L M N O P Q R S T
V Z A B C D E F G H I L M N O P Q R S T U
Z A B C D E F G H I L M N O P Q R S T U V

Si noti che a lettere uguali del messaggio in chiaro corrispondono a lettere cifrate diverse, a seconda del valore della parola chiave, e ciò determina l'appiattimento delle frequenze relative delle lettere del crittogramma.

La decrittazione di un cifrario polialfabetico non è molto più complessa di quella relativa ai cifrari a sostituzione semplice, in quanto supponendo d'aver individuato la lunghezza h del periodo (o la lunghezza della parola di codice nel caso di Vigenère), si tratta di effettuare la forzatura di h cifrari a sostituzione semplice. Il problema è semmai quello di trovare h, e una tecnica generale per risolverlo fu individuata dall'ufficiale prussiano Kasiski, quasi 300 anni dopo l'invenzione del cifrario di Vigenère. Essa si basa sull'osservazione, evidentemente semplice solo a posteriori, che porzioni identiche di messaggio che si trovano a distanze che sono un multiplo del periodo vengono cifrate nello stesso modo. Nel caso dell'esempio relativo alla (7.7) si ha $h = 9$ (lunghezza della parola di codice); se ora la terna "*PER*" si trova nel messaggio ripetuta a una distanza che sia un multiplo di 9, in entrambi i casi viene cifrata con la terna "*CEL*". La strategia è allora quella di cercare stringhe identiche a una stringa prefissata e che nel crittogramma vengono ripetute diverse volte, annotando le distanze reciproche; se troviamo p.es. alcuni spezzoni "*CEL*" all'interno del crittogramma, a distanze $90, 45, 37, 63$, è ragionevole ipotizzare che il periodo sia $9 = \text{MCD}(90, 45, 63)$ e che il "*CEL*" relativo alla distanza 37 derivi invece da un blocco in chiaro e da una chiave diversi. Naturalmente questa prima stima potrebbe dimostrarsi erronea; tuttavia se il crittogramma è abbastanza lungo si

troveranno diverse stringhe ripetute, oltre a *CEL*, con le quali si possono ricavare altre stime. Succederà allora che per un certo numero di stringhe, anche di lunghezze diverse, si troverà sempre il 9 come ipotesi di lavoro, e su questa base si può tentare la decrittazione dei 9 cifrari $S_j, (1 \leq j \leq 9)$ a sostituzione monoalfabetica , cioè quelli con le lettere nelle posizioni $9k + j$ $(k = 0, 1, 2, \ldots)$.

Un altro metodo, ancora più efficace, per individuare la lunghezza del periodo consiste nello sfruttare il cosiddetto *indice di coincidenza* (IC), introdotto da Friedman nel 1920, che misura la probabilità che due lettere prese a caso da un testo cifrato preassegnato siano uguali.

Il valore di IC, che si può calcolare direttamente dal crittogramma, dipende dalla lunghezza h del periodo, e noto il suo valore atteso $IC(h)$ per diversi valori di h si può risalire a h.

Assegnato un alfabeto $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ di cardinalità K per le lettere di un testo cifrato, siano $p_j, 1 \leq j \leq K$, le corrispondenti probabilità a priori che una lettera scelta a caso sia a_j. Il metodo si basa sulla definizione della *misura di convessità* MC associata alle probabilità p_j

$$MC = \sum_{j=1}^{K} \left(p_j - \frac{1}{K} \right)^2 = \sum_{j=1}^{K} p_j^2 - \frac{1}{K} = \nu_\lambda - \frac{1}{K} \tag{7.8}$$

dove

$$\nu_\lambda = \sum_{j=1}^{K} p_j^2 \tag{7.9}$$

MC dipende sensibilmente dal periodo della sostituzione polialfabetica; esso assume il valore minimo (0) in corrispondenza di una distribuzione uniforme $(p_j = 1/K)$ e il valore massimo quando la disomogeneità è massima. Poiché la cifratura polialfabetica tende ad appiattire le frequenze relative delle lettere, è ragionevole supporre che il valore massimo venga ottenuto in corrispondenza di una sostituzione "polialfabetica" di periodo 1, cioè una sostituzione monoalfabetica, mentre il valore minimo (0) si abbia quando $h \to \infty$. Sostituendo i valori riportati nella tabella (7.1) per la lingua italiana, con $K = 21$, si ottengono i seguenti parametri

$$\begin{aligned} &MC_{IT}(1) = \nu_{IT} - \tfrac{1}{K} \approx 0.073 - \tfrac{1}{21} = 0.025 \\ &MC_{IT}(\infty) \approx 0 \end{aligned} \tag{7.10}$$

Disponendo di un valore di MC ricavato direttamente dal crittogramma, e di un suo valore atteso $MC(h)$, sarebbe possibile in linea di principio ricavare h. Ma le p_j non sono note a priori, e bisogna tentare una loro stima. Sia allora

$F = (f_1, f_2, \ldots, f_K)$ il vettore delle frequenze relative che si osservano su un blocco di crittogramma di lunghezza n; si ha $\sum_{j=1}^{K} f_j = n$. Tenuto ora conto che ci sono $\binom{n}{2}$ modi per selezionare una coppia di lettere fra le n del crittogramma e $\binom{f_j}{2}$ per scegliere due lettere uguali a_j, definiamo l'indice di coincidenza IC del crittogramma come

$$IC = \frac{\sum_{j=1}^{K} \binom{f_j}{2}}{\binom{n}{2}} = \frac{\sum_{j=1}^{K} f_j(f_j - 1)}{n(n-1)} \tag{7.11}$$

che costituisce una stima del parametro $\nu_\lambda = \sum_{j=1}^{K} p_j^2$, cioè della probabilità che scegliendo a caso due lettere esse siano uguali.

Ma essendo anche $\nu_\lambda = MC + \frac{1}{K}$ si ricava che IC è anche stima di $MC + \frac{1}{K}$ e dunque varia fra ν_λ (quando $h = 1$) e $\frac{1}{K}$

$$\begin{aligned} IC_{IT}(1) &= \nu_\lambda \approx 0.073 \\ IC_{IT}(\infty) &= \tfrac{1}{21} = 0.048 \end{aligned} \tag{7.12}$$

Si può dimostrare (si veda p.es. [5]) che il valore atteso per $IC(h)$ è dato dall'espressione

$$IC(h) = \frac{1}{h}\frac{n-h}{n-1}\nu_\lambda + \frac{h-1}{h}\frac{n}{n-1}\frac{1}{K} \tag{7.13}$$

che consente di ottenere una tabella, significativa soprattutto per piccoli valori di h, che avrà come estremo inferiore $IC_{IT}(1) \approx 0.073$ ed estremo superiore $IC_{IT}(\infty) = 0.048$. La valutazione dell'indice di coincidenza mediante la (7.11) consente dunque di stimare h, e l'uso di questa informazione congiuntamente al test di Kasiski permette di forzare velocemente il cifrario.

Non si trascuri tuttavia l'importanza concettuale del cifrario a sostituzione polialfabetica; l'aumento della lunghezza della parola chiave offre infatti un incremento della difficoltà nella forzatura, consentendo di aumentare anche la sicurezza del sistema. Le estreme conseguenze di questo ragionamento verranno analizzate nella sezione 7.4.

7.3.3 Cifrario a trasposizione

Assieme alla sostituzione è la più semplice trasformazione che si possa immaginare di effettuare su un testo in chiaro. Il procedimento consiste nel suddividere il messaggio in blocchi di lunghezza p, permutando successivamente le lettere di ciascun blocco secondo uno schema preordinato. Prendendo per esempio $p = 13$, con la permutazione

$$\pi = (3, 12, 6, 10, 9, 2, 7, 1, 4, 11, 5, 8, 13) \tag{7.14}$$

in prima posizione del blocco-crittogramma viene posta la lettera che nel testo in chiaro sta in 3^a posizione, in seconda quella che stava nella 12^a e così via. Di conseguenza $PERMESIVANELLACITTADOLENTE\ldots$, una volta depurato degli spazi viene cifrato come

1 2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13 ...
P E R M E S I V A N E L L A C I T T A D O L E N T E ...

3 12 6 10 9 2 7 1 4 11 5 8 13 3 12 6 10 9 2 7 1 4 11 5 8 13 ...
R L S N A E I P M E E V L I T A E L C D A T N T O E ...

Le chiavi a disposizione sono in numero di $p!$ e quindi il loro numero cresce rapidamente con p. Anche se le frequenze relative delle lettere rimane costante tra testo in chiaro e testo cifrato, le tecniche di forzatura sono del tutto analoghe a quelle statistiche viste precedentemente. In questo caso si tratta di ricorrere alle frequenze dei digrammi, trigrammi ecc. accoppiando lettere che hanno un'elevata probabilità di stare vicine; si pensi p.es. al caso limite, nella lingua italiana, della "q" e della "u" che tendono a diventare "qu" e di conseguenza ogniqualvolta capita una "q" si va subito a cercare la "u" che le corrisponde, e così via.

Per quanto i cifrari appena descritti, singolarmente presi, non abbiano un interesse operativo, essi sono importanti perchè costituiscono due tipi di trasformazioni di base la quali, sinergicamente sfruttate, possono portare a dei cifrari estremamente sicuri. Shannon usò i termini *confusione* e *diffusione* per esprimere l'effetto delle due trasformazioni. Nella sostituzione si tende a rendere più complesso il legame tra chiave e testo cifrato, e questo corrisponde all'introduzione di "confusione"; viceversa per "diffusione" s'intende un riarrangiamento delle lettere del testo in chiaro in modo che ogni ridondanza dello stesso venga per così dire distribuita sull'intero crittogramma, e questo corrisponde a una trasposizione. Il dosaggio sapiente di confusione e diffusione nei cifrari di tipo composto incrementa in modo rilevante la sicurezza degli stessi. È infatti evidente che questi schemi possono essere usati in cascata, in modo che il testo cifrato del primo cifrario diventi testo in chiaro per il cifrario successivo. Il fattore che limita una complicazione smisurata della struttura di base del cifrario composto, facile tentazione per un crittografo dilettante, è al solito la circostanza che il progettista del cifrario deve poi essere in grado di dimostrarne la sicurezza, convincendo l'acquirente che non ci sono debolezze occulte che potrebbero derivare anche da trasformazioni involutive, che alla fine di un lungo e complicato processo potrebbero lasciare il messaggio quasi inalterato.

7.3.4 Cifrari a rotore

Il cifrario a sostituzione polialfabetica conseguì i suoi massimi successi nei primi anni del 900, grazie alle macchine cifranti a *rotori*, che consentivano una meccanizzazione nella costruzione delle successive permutazioni. Il principio sul quale si basano queste macchine, la più famosa delle quali fu senza dubbio la tedesca *Enigma*, è illustrato nella figura 7.4. In sostanza il sistema è costituito da una certo numero di rotori cilindrici che possono ruotare in moto odometrico sopra un asse (come analogia si pensi ai cilindri del contachilometri meccanico delle autovetture). Ciascun rotore dispone sui due lati di 26 contatti (uno per ciascuna lettera dell'alfabeto tedesco) e i contatti sono in collegamento elettrico internamente al cilindro in modo da realizzare una tra le $26!$ possibili permutazioni. Di conseguenza entrando nel rotore sul contatto relativo a una certa lettera, la continuità elettrica stabilisce l'uscita in corrispondenza della lettera associata dalla permutazione. Poiché i contatti di uscita di un cilindro sono premuti

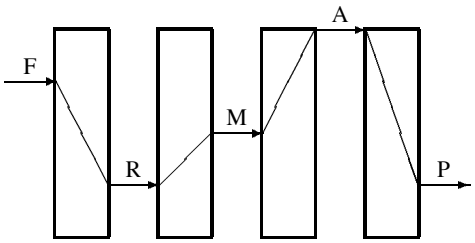


Figura 7.4 Permutazioni nei rotori della macchina Enigma.

sui contatti d'ingresso del cilindro successivo, la lettera in uscita dalla macchina è quella che deriva dalla concatenazione di un numero di permutazioni pari al numero di cilindri (nel caso dell'Enigma ci furono diverse versioni, l'ultima delle quali disponeva di 5 cilindri). La macchina è dotata di una tastiera e di un pannello luminoso; premendo il tasto corrispondente a una certa lettera di testo in chiaro si accende la luce relativa alla lettera del testo cifrato; la pressione determina inoltre la rotazione di un passo del cilindro di destra, innescando così una nuova permutazione per la lettera in chiaro successiva. Dopo 26 passi l'ultimo cilindro acquisisce la posizione iniziale, determinando il movimento di un passo del penultimo cilindro, e così via. Se ci sono 5 rotori il periodo della chiave è pari a $26^5 \approx 11.9\ 10^6$. La macchina è corredata anche di banchi per delle sostituzioni suppletive, realizzate mediante collegamenti esterni, nonché di una dotazione di rotori supplementari da poter avvicendare a quelli montati per cambiare la struttura di base delle permutazioni.

Il punto di forza dei sistemi a rotore fu l'aumento della lunghezza della parola chiave, che determinò notevoli difficoltà per la forzatura del cifrario ; questa avvenne comunque nel 1943, per opera di un gruppo di matematici capeggiati da Alan Turing (il padre del modello computazionale denominato *macchina di Turing*). È verosimile che la forzatura dell'Enigma, all'insaputa dei Tedeschi che continuarono imperterriti a trasmettere messaggi cifrati mediante questa macchina (da loro ritenuta assolutamente inviolabile!), determinò un nuovo assetto nelle vicende belliche, che divenne in seguito molto più favorevole agli Alleati.

7.4 Cifrari ideali e cifrari perfetti

La struttura dei cifrari classici, a trasposizione e a sostituzione, ci è utile per mettere in luce l'approccio shannoniano alla crittografia, e in particolare per individuare gli elementi sui quali far leva al fine di migliorare le prestazioni dei cifrari classici.

L'analisi delle prestazioni dei sistemi crittografici può trarre beneficio dall'impiego delle misure d'informazione studiate nel capitolo 2. Poiché gli oggetti di nostro interesse sono rispettivamente il messaggio m, la chiave k e il crittogramma c, introduciamo le corrispondenti variabili aleatorie M, K e C per descriverne il comportamento statistico. Ricordiamo la (7.1), che nel nostro caso possiamo scrivere come

$$C = Cif(M, K) \qquad\qquad M = Dec(C, K) \tag{7.15}$$

Cifratura e decifrazione sono dunque operazioni deterministiche, legate in modo funzionale rispettivamente a M, K e a C, K.

Immaginiamo ora di aver intercettato una porzione di crittogramma; poiché esso deriva da un'operazione di cifratura, il crittogramma contiene implicitamente dell'informazione sul messaggio trasmesso e sulla chiave usata. Ha senso tentare una valutazione di quest'informazione, o meglio una valutazione dell'incertezza a priori. Poiché l'entropia misura l'incertezza a priori associata alla realizzazione di una certa variabile aleatoria, le prime due tra le quantità

$$H(M/C) \qquad H(K/C) \qquad H(K/M, C) \tag{7.16}$$

esprimono la quantità media d'incertezza su M e K condizionata alla conoscenza del crittogramma C, mentre l'ultima corrisponde all'incertezza su K dopo aver effettuato un attacco crittanalitico con testo in chiaro (noti cioè un crittogramma c e il corrispondente messaggio m). Queste entropie condizionate sono delle *equivocazioni* e in un sistema crittografico sicuro devono essere tutte e tre di valore elevato.

Notiamo ora che

$$\begin{aligned} H(K, M/C) &= H(K/C) + H(M/K, C) \\ &= H(M/C) + H(K/M, C) \end{aligned}$$

e poiché $H(M/K, C) = 0$, giacché noti chiave e crittogramma si risale al messaggio in modo univoco mediante l'operazione deterministica di cifratura, si ottiene

$$H(K/C) = H(M/C) + H(K/M, C) \qquad \text{cioè} \tag{7.17}$$

$$H(K/C) \geq H(M/C) \tag{7.18}$$

che è la *disuguaglianza fondamentale della crittologia*; essa esprime il fatto che l'incertezza media sulla chiave noto il crittogramma è sempre maggiore o uguale all'incertezza sul messaggio (noto il crittogramma). In altre parole individuare la chiave è mediamente più difficile che individuare il messaggio. Ciò deriva dal fatto che $H(K/M, C)$ non è nullo (anzi, dev'essere elevato come si diceva poc'anzi).

Come esempio immediato si pensi al cifrario a trasposizione sotto riportato, dove MARTE e $RMEAT$ sono rispettivamente il messaggio inserito nella macchina cifrante e il crittogramma ottenuto

$$\begin{array}{ccccc} \text{M} & \text{A} & \text{R} & \text{T} & \text{E} \\ & & & & \\ R & M & E & A & T \end{array} \tag{7.19}$$

La chiave viene individuata univocamente, e corrisponde alla permutazione $\pi = (3, 1, 5, 2, 4)$. Se però inseriamo nella stessa macchina la parola MAREA, il crittogramma che ne esce, cioè $RMAAE$

$$\begin{array}{ccccc} \text{M} & \text{A} & \text{R} & \text{E} & \text{A} \\ & & & & \\ R & M & A & A & E \end{array} \tag{7.20}$$

non ci consente d'individuare la chiave in modo univoco, visto che sono possibili tanto la $\pi_1 = (3, 1, 5, 2, 4)$ quanto la $\pi_2 = (3, 1, 2, 5, 4)$.

Con riferimento alle equivocazioni (7.16), la situazione migliore che si possa determinare per un cifrario è quella in cui il crittogramma *non porta alcuna informazione* su chiave e messaggio; nel primo caso si parla di *cifrario ideale* e nel secondo di *cifrario perfetto*. Queste condizioni si possono esprimere con le seguenti eguaglianze

$$\begin{aligned} &\text{Cifrario ideale} \quad H(K/C) = H(K) \implies I(K \wedge C) = 0 \\ &\text{Cifrario perfetto} \quad H(M/C) = H(M) \implies I(M \wedge C) = 0 \end{aligned} \tag{7.21}$$

L'interpretazione dell'idealità e della perfezione fatta attraverso la mutua informazione risulta particolarmente efficacie, poiché in tal modo si stabilisce che crittogramma e chiave (messaggio) non si scambiano informazione.

7.4.1 Un esempio di cifrario ideale

Diamo ora un esempio di comportamento ideale per un cifrario; a tal fine consideriamo un cifrario a trasposizione, di periodo p, che effettui la cifratura dei messaggi che escono da una sorgente stazionaria e senza memoria. Supponiamo inoltre che si possano usare tutte le possibili $p!$ chiavi, scegliendole sulla base di una distribuzione uniforme. In queste condizioni vale il seguente

Teorema 7.2. *In un cifrario a trasposizione per una sorgente dei messaggi stazionaria e senza memoria vale la relazione*

$$H(K/C^{np}) = H(K) \tag{7.22}$$

dove C^{np} è una v.a. associata a n blocchi di lunghezza p del crittogramma.

Dim. Si deve provare che $p(k, \boldsymbol{c}) = p(k) \cdot p(\boldsymbol{c})$ (usiamo i simboli $\boldsymbol{c}$ e $\boldsymbol{m}$ per ricordare che crittogramma e messaggio sono in realtà vettori) oppure, tenendo conto dell'ipotesi di chiavi equiprobabili, che

$$p(k, \boldsymbol{c}) = \frac{1}{p!}\, p(\boldsymbol{c})$$

visto che $p(k) = 1/p!$. Poiché $\boldsymbol{m} = Dec(\boldsymbol{c}, k)$ è deterministica, si ha $p(\boldsymbol{m}/k, \boldsymbol{c}) = 1$, e dunque si può scrivere, tenendo conto che anche $\boldsymbol{c} = Cif(\boldsymbol{m}, k)$ è deterministica

$$\begin{aligned} p(k, \boldsymbol{c}) &= p(\boldsymbol{m}/k, \boldsymbol{c}) \cdot p(k, \boldsymbol{c}) = p(\boldsymbol{m}, k, \boldsymbol{c}) = \\ &= p(\boldsymbol{m}, k) \cdot p(\boldsymbol{c}/\boldsymbol{m}, k) = p(\boldsymbol{m}, k) = p(k)p(\boldsymbol{m}/k) = \\ &= \frac{1}{p!}\, p(\boldsymbol{m}(k, \boldsymbol{c})) \end{aligned}$$

dove l'ultima uguaglianza deriva dal fatto che messaggio e chiave sono indipendenti. La notazione $p(\boldsymbol{m}(k, \boldsymbol{c}))$ ci ricorda inoltre che la probabilità in questione è quella relativa al messaggio $\boldsymbol{m}$ che si ottiene dalla decifrazione del crittogramma $\boldsymbol{c}$ effettuata con la chiave k. Di conseguenza per giungere alla tesi basta provare che $p(\boldsymbol{m}(k, \boldsymbol{c})) = p(\boldsymbol{c})$. L'uguaglianza è ovvia nelle condizioni di sorgente stazionaria e senza memoria, poiché il vettore $\boldsymbol{c}$ si ottiene da $\boldsymbol{m}$ mediante una semplice permutazione delle lettere, che non cambia la probabilità dell'ennupla relativa, che come noto dipende solo dalla sua composizione □

Il fatto che il cifrario a trasposizione sia ideale non implica però che sia $H(K/M^{np}, C^{np}) = 0$, come appare del resto evidente dai commenti relativi alla (7.20). È tuttavia ragionevole ipotizzare che l'equivocazione sulla chiave, noti messaggio e crittogramma, tenda a diventare sempre più piccola al crescere di n. Del resto se si suppone che le due parole in chiaro relative alle (7.20)

e (7.19) si presentino in quest'ordine, e si disponga dei corrispondenti crittogrammi, se in corrispondenza della prima (MAREA) rimane un'incertezza residua sulla chiave, all'apparire della seconda (MARTE) l'incertezza viene sciolta, giacché si riesce a escludere la $\pi_2 = (3,1,2,5,4)$, rimanendo con la sola $\pi_1 = (3,1,5,2,4)$.

Teorema 7.3. *In un cifrario a trasposizione per una sorgente stazionaria e senza memoria si ha* $H(K/M^{np}, C^{np}) \to 0$ *quando* $n \to \infty$.

Dim. La stima della chiave fallisce quando ci sono coppie di lettere uguali in due posizioni arbitrarie i e j, con $i \neq j$, all'interno del primo periodo p; chiamiamo tale evento sfavorevole $u(i,j)$. Se dunque p_h è la probabilità associata alla lettera a_h di un alfabeto K-ario, la probabilità di tale evento è pari a

$$\Pr\{u(i,j)\} = p(i,j) = \sum_{h=1}^{K} p_h^2$$

mentre la probabilità globale dell'evento che riguarda l'uguaglianza di una qualunque tra le $\binom{p}{2} = p(p-1)/2$ posizioni possibili è pari a

$$\Pr\{u\} = \Pr\{u(1,2) \cup u(1,3) \cup \ldots \cup u(p-1,p)\} \leq \frac{p(p-1)}{2} \sum_{h=1}^{K} p_h^2$$

che viene limitata tenendo conto che gli eventi non sono necessariamente indipendenti (in caso d'indipendenza varrebbe l'uguaglianza).

D'altra parte, acquisendo n blocchi successivi del crittogramma la conoscenza della chiave viene negata solo se esiste almeno una coppia di posizioni i e j, con $i \neq j$, per le quali si hanno lettere uguali in corrispondenza di *tutti* gli n blocchi. La probabilità di tale evento sfavorevole di ordine n è

$$\Pr\{u^{(n)}(i,j)\} = p^{(n)}(i,j) = \left(\sum_{h=1}^{K} p_h^2 \right)^n$$

mentre quella dell'evento globale di ordine n è limitata superiormente dalla

$$\Pr\{u^{(n)}\} \leq \frac{p(p-1)}{2} \left(\sum_{h=1}^{K} p_h^2 \right)^n \tag{7.23}$$

Ma essendo $\sum_{h=1}^{K} p_h^2 < \left(\sum_{h=1}^{K} p_h \right)^2 = 1$ se P è non degenere, la limitazione superiore della (7.23) tende a zero quando $n \to \infty$. Tenendo conto che l'evento $u^{(n)}$ corrisponde a un fallimento nella stima esatta della chiave, possiamo scrivere

$$\Pr\{u^{(n)}\} = \Pr(K \neq \widehat{K}) = \epsilon \longrightarrow 0 \qquad \text{quando} \qquad n \to \infty \tag{7.24}$$

Dunque la probabilità di effettuare una stima sbagliata tende a zero con n. Avendo la Pr$(K \neq \widehat{K})$ possiamo ricorrere alla disuguaglianza di Fano (2.30); tenendo conto anche della (2.22) e del fatto che la stima $\widehat{k}$ si ottiene da una procedimento deterministico $\widehat{k} = f(\boldsymbol{m}, \boldsymbol{c})$, che ha come variabili $\boldsymbol{m}$ e $\boldsymbol{c}$, possiamo scrivere la seguente relazione

$$H(K/M^{np}, C^{np}) \leq H(K/f(M^{np}, C^{np})) = H(K/\widehat{K}) \leq$$
$$\leq h(\epsilon) + \epsilon \log(p! - 1) \longrightarrow 0 \tag{7.25}$$

che tende a 0 con ϵ quando $n \to \infty$. □

Il cifrario a trasposizione è dunque ideale, e dovrebbe fornire delle buone prestazioni, ma è molto debole dal punto di vista degli attacchi crittanalitici con testo in chiaro per effetto della (7.25). Si tenga inoltre conto che gli attacchi statistici sono in questo caso efficaci, e in pratica si riesce a forzare il cifrario anche senza attacchi con testo in chiaro. Dalla (7.17) si ricava inoltre

$$H(K) \longrightarrow H(M^{np}/C^{np})$$

e dunque l'incertezza sulla chiave collassa sull'equivocazione del messaggio noto il crittogramma.

7.4.2 Un esempio di cifrario perfetto

Riprendiamo ora la definizione di cifrario perfetto (7.21); essa implica che sia

$$p(\boldsymbol{m}/\boldsymbol{c}) = p(\boldsymbol{m}) \tag{7.26}$$

per ogni coppia $\boldsymbol{m}$ e $\boldsymbol{c}$. D'altra parte, per il *teorema di Bayes* [34] possiamo scrivere

$$p(\boldsymbol{m}, \boldsymbol{c}) = p(\boldsymbol{c})p(\boldsymbol{m}/\boldsymbol{c}) = p(\boldsymbol{m})p(\boldsymbol{c}/\boldsymbol{m}) \tag{7.27}$$

che porta al seguente

Lemma 7.1. *Condizione necessaria e sufficiente per la perfezione di un cifrario è che valga la*

$$p(\boldsymbol{c}/\boldsymbol{m}) = p(\boldsymbol{c}) \tag{7.28}$$

per ogni messaggio $\boldsymbol{m}$ e crittogramma $\boldsymbol{c}$.

Il lemma impone che per qualunque coppia di messaggi m_i, m_j e per ogni crittogramma c, le probabilità di tutte le chiavi che trasformano m_i in c o m_j in c devono essere uguali, visto che $p(c/m_i) = p(c) = p(c/m_j)$; se le chiavi sono equiprobabili l'uguaglianza si estende al numero delle chiavi.

Questo lemma ha un ruolo decisivo nello stabilire le condizioni da imporre per garantirsi la perfezione, che sono come vedremo particolarmente gravose. Vale infatti il seguente

Teorema 7.4. *Se $\mathcal{M}$ è lo spazio dei messaggi di un cifrario perfetto, $\mathcal{C}$ lo spazio dei crittogrammi e $\mathcal{K}$ quello delle chiavi, vale la seguente disuguaglianza*

$$|\mathcal{K}| \geq |\mathcal{M}| \tag{7.29}$$

Dim. Dimostreremo il teorema in due passi, verificando prima che $|\mathcal{K}| \geq |\mathcal{C}|$ e poi che $|\mathcal{C}| \geq |\mathcal{M}|$.

La prima disuguaglianza si dimostra fissando m; poiché $p(c/m) = p(c) > 0$ per l'ipotesi di cifrario perfetto, per ogni crittogramma c esiste una chiave k tale che $c = Cif(m, k)$, e dunque $|\mathcal{K}| \geq |\mathcal{C}|$.

Per la seconda si consideri che, fissata una certa chiave k, per ogni messaggio m esiste un crittogramma c tale che $c = Dec(m, k)$ in quanto, fissata la chiave k, $Cif(m_i, k)$ e $Cif(m_j, k)$ sono crittogrammi diversi. □

Nel caso in cui le cardinalità degli spazi dei messaggi, crittogrammi e chiavi coincidano, dal lemma (7.1) e dal teorema 7.4 si giunge immediatamente al seguente

Teorema 7.5. *Se $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$ un cifrario è perfetto se e solo se c'è esattamente una chiave che trasforma ciascun messaggio in ciascun crittogramma e tutte le chiavi sono equiprobabili*.

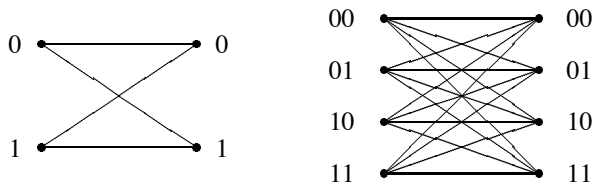


Figura 7.5 Due esempi di cifrari perfetti.

Le implicazioni pratiche dei teoremi 7.4 e 7.5 sono molto pesanti, poiché per raggiungere la perfezione è necessario scegliere (a caso) una fra le possibili chiavi, effettuare la cifratura e ripetere la procedura per il messaggio successivo. Se

il messaggio è un blocco di lunghezza n, anche la lunghezza della chiave deve essere quantomeno n, e dunque *la chiave ha una lunghezza almeno pari a quella del messaggio*. In queste condizioni l'ipotesi di poter trasmettere la chiave su un canale sicuro, così come si era supposto all'inizio, viene completamente a cadere, poiché in questo caso si potrebbe trasmettere direttamente il messaggio. In figura 7.5 vengono illustrati due esempi di cifrari perfetti, nei quali si nota che cardinalità dei messaggi e delle chiavi coincidono. Il più semplice, noto col nome di *one-time pad*, trova attuazione nello schema di figura 7.6 (nella versione binaria). L'alfabeto di messaggio, chiave e crittogramma è binario, e ciascun bit

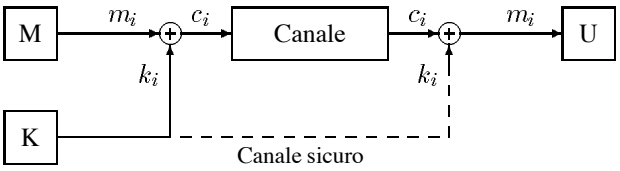


Figura 7.6 Cifrario *one-time pad*.

c_i del crittogramma si ottiene sommando bit per bit $\mod 2$ il bit del messaggio col bit relativo alla chiave

$$c_i = m_i \oplus k_i \tag{7.30}$$

Per garantire la perfezione la sorgente delle chiavi k_i dev'essere aleatoria, stazionaria e senza memoria con distribuzione uniforme. Anche se la perfezione è implicitamente affermata dal teorema 7.5, dimostriamo che vale la (7.21)

Teorema 7.6. *Se $\boldsymbol{M}$, $\boldsymbol{C}$ e $\boldsymbol{K}$ sono vettori di variabili aleatorie associate ai blocchi di dimensione n e relativi a $\boldsymbol{m}$, $\boldsymbol{c}$ e $\boldsymbol{k}$, per un cifrario one-time pad vale la seguente relazione*

$$H(\boldsymbol{M}/\boldsymbol{C}) = H(\boldsymbol{M}) \tag{7.31}$$

Dim. Si deve dimostrare che $p(\boldsymbol{m}/\boldsymbol{c}) = p(\boldsymbol{m})$. Si noti che in questo caso, noti $\boldsymbol{m}$ e $\boldsymbol{c}$, la chiave $\boldsymbol{k}$ si ricava deterministicamente come $\boldsymbol{k} = \boldsymbol{m} \oplus \boldsymbol{c}$, e dunque $p(\boldsymbol{k}/\boldsymbol{m}, \boldsymbol{c}) = 1$. Possiamo allora scrivere

$$\begin{aligned} p(\boldsymbol{m}, \boldsymbol{c}) &= p(\boldsymbol{m}, \boldsymbol{c}) p(\boldsymbol{k}/\boldsymbol{m}, \boldsymbol{c}) = p(\boldsymbol{m}, \boldsymbol{k}, \boldsymbol{c}) = \\ &= p(\boldsymbol{k}) p(\boldsymbol{m}/\boldsymbol{k}) p(\boldsymbol{c}/\boldsymbol{m}, \boldsymbol{k}) = \frac{1}{2^n} p(\boldsymbol{m}) \end{aligned}$$

dove l'ultima uguaglianza deriva dal fatto che chiave e messaggio sono indipendenti e che $\boldsymbol{c} = \boldsymbol{m} \oplus \boldsymbol{k}$. Per la probabilità del crittogramma si ricava allora

$$p(\boldsymbol{c}) = \sum_{\boldsymbol{m}} p(\boldsymbol{m}, \boldsymbol{c}) = \frac{1}{2^n} \sum_{\boldsymbol{m}} p(\boldsymbol{m}) = \frac{1}{2^n}$$

Si ha dunque

$$\frac{p(\boldsymbol{m}, \boldsymbol{c})}{p(\boldsymbol{c})} = \frac{p(\boldsymbol{c}) p(\boldsymbol{m}/\boldsymbol{c})}{p(\boldsymbol{c})} = p(\boldsymbol{m}/\boldsymbol{c}) \qquad \text{ma anche}$$
$$\frac{p(\boldsymbol{m}, \boldsymbol{c})}{p(\boldsymbol{c})} = \frac{\frac{1}{2^n} p(\boldsymbol{m})}{\frac{1}{2^n}} = p(\boldsymbol{m})$$

cioè la tesi □

Il cifrario *one-time pad* è dunque perfetto, anche se sembra essere particolarmente debole dal punto di vista degli attacchi crittanalitici con testo in chiaro, poiché $H(\boldsymbol{k}/\boldsymbol{m}, \boldsymbol{c}) = 0$. Si noti però che la conoscenza della chiave passata $\boldsymbol{k}$ non fornisce alcuna informazione sulla chiave futura $\boldsymbol{k_f}$ che verrà generata dalla sorgente aleatoria delle chiavi. Di conseguenza possiamo scrivere

$$\begin{aligned} H(\boldsymbol{k_f}/\boldsymbol{m}, \boldsymbol{c}) &= H(\boldsymbol{k_f}) \\ H(\boldsymbol{k_f}/\boldsymbol{c}) &= H(\boldsymbol{k_f}) \end{aligned} \tag{7.32}$$

La prima delle (7.32) stabilisce la perfetta resistenza nei confronti degli attacchi con testo in chiaro, mentre la seconda implica che il cifrario, oltre che essere perfetto è anche ideale. Il crittogramma non fornisce dunque alcuna informazione né sul messaggio né sulla chiave e il sistema *non è soggetto a forzature*.

La resistenza incondizionata a qualunque forma di attacco esibita dal sistema *one-time pad* richiede però un prezzo estremamente alto, e cioè che la chiave sia lunga almeno quanto il crittogramma. Ciò rende il sistema difficile da utilizzare, a meno che i due utenti non s'incontrino e generino concordemente una chiave da utilizzare in un secondo momento, quando ciascuno ha raggiunto la propria sede. Non si trascuri tuttavia l'impiego del sistema *one-time pad* secondo il protocollo stabilito dalla (7.2), che richiede però tre trasmissioni sul canale.

Nonostante ciò il cifrario *one-time pad* è tuttavia il più *economico* tra i cifrari perfetti, poiché vale la seguente relazione

$$H(\boldsymbol{K}) \geq H(\boldsymbol{K}/\boldsymbol{C}) \geq H(\boldsymbol{M}/\boldsymbol{C}) = H(\boldsymbol{M}) \tag{7.33}$$

(la seconda disuguaglianza è quella fondamentale della crittologia (7.18) e l'ultima uguaglianza deriva dalla perfezione del cifrario), che è un altro modo per esprimere il fatto che la lunghezza della chiave supera quella del messaggio. Nel sistema *one-time pad* tale relazione vale come uguaglianza.

7.5 Cifrari aleatori e distanza di unicità

Abbiamo visto che la condizione di perfezione impone una cardinalità dello spazio delle chiavi pari almeno alla cardinalità dei messaggi. Ciò implica che persino un cifrario a rotazione è perfetto se lo si usa su un'unica lettera. Se però la lunghezza del messaggio aumenta bisogna contemporaneamente incrementare la lunghezza della chiave, altrimenti il sistema diventa rapidamente insicuro. Nell'esempio di figura 7.7 si vede lo sviluppo del fenomeno relativo a un cifrario a sostituzione semplice, nel quale a seguito del passaggio da $n = 1$ a $n = 2$ e $n = 3$, senza un contemporaneo incremento dello spazio delle chiavi l'acquisizione del crittogramma porta sempre più informazione sul messaggio, riducendo la cardinalità relativa delle classi residue di messaggi e crittogrammi. Tale fenomeno può essere analizzato, dal punto di vista quantitativo, studiando

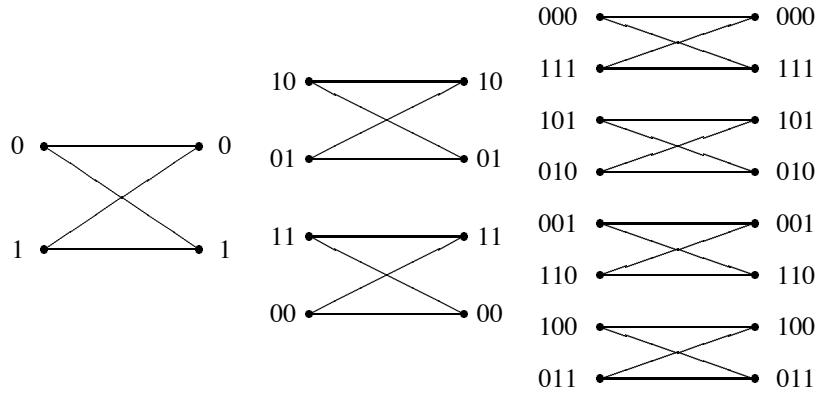


Figura 7.7 Classi residue per $n = 1, 2, 3$ in un cifrario a sostituzione semplice.

l'andamento delle mutue informazioni $I(K \wedge C)$ e $I(M \wedge C)$, che misurano la quantità d'informazione su M e K che si acquisisce mediamente da C. In generale l'acquisizione di un'ulteriore lettera di crittogramma non può aumentare l'incertezza che si ha sul messaggio e sulla chiave. Ciò si esprimere con le seguenti disuguaglianze

$$\begin{aligned} H(K/\boldsymbol{C^{n+1}}) &\leq H(K/\boldsymbol{C^n}) \\ H(\boldsymbol{M^h}/\boldsymbol{C^{n+1}}) &\leq H(\boldsymbol{M^h}/\boldsymbol{C^n}) \end{aligned} \tag{7.34}$$

con ovvio significato dei simboli.

Per mettere in luce la variazione di $I(K \wedge \boldsymbol{C^n})$ con n possiamo calcolare l'entropia congiunta $H(\boldsymbol{M^n}, \boldsymbol{C^n}, K)$, tenendo conto che cifratura e decifrazione implicano rispettivamente $H(\boldsymbol{C^n}/\boldsymbol{M^n}, K) = 0$ e $H(\boldsymbol{M^n}/\boldsymbol{C^n}, K) = 0$. Si

ricava dunque

$$H(\boldsymbol{M^n}, \boldsymbol{C^n}, K) = H(\boldsymbol{M^n}, K) + H(\boldsymbol{C^n}/\boldsymbol{M^n}, K) = H(\boldsymbol{M^n}, K)$$
$$= H(\boldsymbol{C^n}, K) + H(\boldsymbol{M^n}/\boldsymbol{C^n}, K) = H(\boldsymbol{C^n}, K)$$

che implica, tenendo conto che chiave e messaggio sono indipendenti

$$H(\boldsymbol{M^n}, K) = H(\boldsymbol{C^n}, K)$$
$$H(\boldsymbol{M^n}) + H(K) = H(\boldsymbol{C^n}) + H(K/\boldsymbol{C^n}) \tag{7.35}$$

Dalla (7.35) si ricava per confronto che

$$I(K \wedge \boldsymbol{C^n}) = H(K) - H(K/\boldsymbol{C^n}) = H(\boldsymbol{C^n}) - H(\boldsymbol{M^n}) \tag{7.36}$$

Un calcolo diretto di questa quantità non è praticabile; tuttavia sarebbe utile avere quantomeno una limitazione superiore. Per trovarla supponiamo che l'alfabeto $\mathcal{A}$ dei crittogrammi sia lo stesso di quello dei messaggi. Un crittogramma di lunghezza n può assumere, a priori, uno tra $|\mathcal{A}|^n$ valori, e per la sua entropia possiamo scrivere

$$H(\boldsymbol{C^n}) \leq \log |\mathcal{A}|^n = nR_0 \qquad \text{con} \qquad R_0 = \log |\mathcal{A}| \tag{7.37}$$

che è una limitazione rozza, ma sufficiente per i nostri scopi. Per quanto riguarda invece l'entropia sul messaggio possiamo ricorrere, per n sufficientemente grande, all'*entropia media per lettera*

$$R_n = \frac{H(\boldsymbol{M^n})}{n} \tag{7.38}$$

già incontrata nella (3.7). Per questa quantità sono generalmente disponibili delle tabelle con i valori relativi alle varie lingue naturali, che ci consentono di effettuare l'approssimazione $H(\boldsymbol{M^n}) \approx nR_\infty$ quando n è sufficientemente grande. Ponendo per semplicità di scrittura $R_\infty = R$, possiamo allora riscrivere la (7.36) come

$$I(K \wedge \boldsymbol{C^n}) = H(K) - H(K/\boldsymbol{C^n}) \leq n(R_0 - R) \tag{7.39}$$

dove alla quantità $D = R_0 - R$ si attribuisce il nome di *ridondanza media per lettera* (nella lingua usata). La (7.39) si può riassumere col seguente

Teorema 7.7. *La quantità d'incertezza rimossa su K dalla conoscenza di n lettere del crittogramma non può superare la ridondanza di n lettere del messaggio.*

Il teorema pone un'importante limitazione normativa all'informazione acquisita sulla chiave disponendo del solo crittogramma.

n	1	2	3	4	5	6	7	8	9	10
R_n	4.003	3.692	3.462	3.236	2.964	2.651	2.344	2.073	1.843	1.652
D_n	0.389	0.700	0.930	1.156	1.428	1.741	2.048	2.319	2.549	2.740

Tabella 7.2 Entropia e ridondanza media per lettera in Italiano (da "I Promessi Sposi").

Esempio 7.2. Consideriamo il caso della lingua italiana, per la quale $R_0 = \log 21 = 4.392$. Sempre da "I Promessi Sposi" sono stati ricavati i seguenti valori per l'entropia media per lettera fino a $n = 10$ Dalla relazione $D_n = R_0 - R_n$ si ricava infine la ridondanza media per lettera, che cresce fino a un valore di saturazione che si stima essere pari a $D_\infty \approx 3.1$. In queste condizioni la ridondanza percentuale passa da meno del 10% per $n = 1$ a quasi il 70% quando $n \to \infty$. Il valore asintotico della ridondanza è abbastanza costante rispetto alla lingua scelta, il che suggerisce che la ridondanza media per lettera delle lingue naturali sia sostanzialmente costante.

L'acquisizione di porzioni sempre più grandi di crittogramma rimuove via via incertezza sul messaggio, consentendo delle stime sempre più precise sullo stesso, anche senza effettuare un attacco con testo in chiaro.

L'esempio successivo (adattato da [5]) mette in forte evidenza il fenomeno.

Esempio 7.3. Si supponga di ricevere porzioni sempre più grandi di un crittogramma che deriva da una cifratura a rotazione. Il crittogramma sia per esempio *oapsgz* . . . . La prima lettera ricevuta è "*o*", e sulla base di quest'unica informazione la stima più ragionevole che si possa fare è che essa derivi da una "*e*", visto che nella tabella 7.1 delle frequenze relative in Italiano è la lettera più frequente. Ciò porta a stimare una rotazione $j = 8$. Con l'acquisizione della seconda lettera la stima si modifica, poiché la coppia "*ep*" non è la più probabile a posteriori, e subentra la "*al*", che porta a $j = 12$. Le probabilità a posteriori della tabella si ottengono dividendo le probabilità assolute di ciascuna n-pla per la somma delle probabilità delle varie n-ple possibili. Ricevendo la terza lettera bisogna aggiornare nuovamente la stima di j, poiché la terna più verosimile diventa $j = 13$ che deriva da "*zia*". Per $n = 4$ si ha sempre come miglior stima una rotazione $j = 13$ ("*ziad*"), a pari merito con $j = 12$ ("*albe*"). La quinta lettera di crittogramma consente però di ricavare in modo univoco la chiave, poiché alla rotazione $j = 12$, che corrisponde alla stringa *"alber"*, viene attribuita tutta la probabilità .

Dall'esempio appare chiaro che l'acquisizione di una porzione sufficiente di crittogramma consente prima o poi di forzare il cifrario, riconoscendo la chiave corretta. Il meccanismo consiste essenzialmente nello scartare le chiavi che portano a messaggi che sono poco (o per nulla) verosimili, e tale tecnica porta a restringere il numero di chiavi attendibili man mano che la lunghezza del crittogramma acquisito aumenta. È ragionevole attendersi che esista una sorta

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
ugvbof	0.03569	0.00821			
vhzcpg	0.02305	0.00102			
ziadqh	0.00760	0.04503	**0.24012**	**0.4**	
alberi	0.11512	**0.21555**	0.02961	**0.4**	**1**
bmcfsl	0.00973				
cndgtm	0.04692	0.00004			
doehun	0.03731	0.11826	0.07895		
epfivo	**0.12059**	0.08846			
fqglzp	0.01052				
grhmaq	0.01713	0.02635			
hsinbr	0.01352	0.00094	0.01645		
itlocs	0.09530	0.09565			
lumpdt	0.05567	0.04437	0.03289		
mvnqeu	0.02365	0.00011			
nzorfv	0.07292	0.04619	0.33388	0.2	
oapsgz	0.09640	0.08733	0.14474		
pbqtha	0.02967				
qcruib	0.00779				
rdsvlc	0.06599	0.03674			
setzmd	0.05459	0.18564	0.12336		
tfuane	0.06083	0.00011			

di lunghezza limite, superata la quale non si ha più alcuna incertezza sulla chiave. Tale aspetto può essere studiato anche in modo quantitativo, ricorrendo a un modello generale di cifrario introdotto da Hellman e noto col nome di *cifrario aleatorio*.

Def. 7.3. *Si definisce aleatorio un cifrario per il quale valgono le seguenti proprietà*

1. *Il numero totale di messaggi di lunghezza n è pari a*

$$T = |\mathcal{A}|^n = 2^{nR_0}$$

2. *I T messaggi possibili si dividono in due classi:*

messaggi significativi, *caratterizzati da una probabilità a priori elevata e (praticamente) uniforme; essi sono in numero di*

$$S = 2^{nR}$$

dove R è l'entropia media per lettera definita dalla (7.38);

messaggi non significativi, *che sono tutti i rimanenti $T - S$, caratterizzati da probabilità a priori trascurabili*

3. *Ci sono $|\mathcal{K}|$ chiavi equiprobabili, e si assume che nel grafo che rappresenta il cifrario, i $|\mathcal{K}|$ archi di ciascun crittogramma si congiungano in modo aleatorio con i messaggi possibili.*

Lo schema di cifrario aleatorio rappresenta un modello generale di cifratura, nel quale l'aleatorietà viene introdotta essenzialmente per poter effettuare delle stime in termini probabilistici riguardo gli eventi favorevoli al crittanalista.

Oss. 7.2. La classe dei messaggi significativi consiste di tutti i messaggi che in una certa lingua naturale sono dotati di "senso compiuto", ed è del tutto analoga, dal punto di vista concettuale, alle sequenze tipiche di cui alla sezione 3.3. In entrambi i casi la suddivisione è tra sequenze che la sorgente può emettere e sequenze che non sono verosimili per la sorgente in oggetto. Nel caso delle sequenze tipiche la sorgente è stazionaria e senza memoria; qui invece la sorgente genera sequenze in una lingua naturale.

Se il cifrario è aleatorio, per ogni crittogramma assegnato la probabilità di ottenere una *decrittazione significativa* a partire da una chiave qualunque vale

$$\frac{S}{T} = 2^{-n(R_0 - R)} = 2^{-nD}$$

Poiché ci sono $|\mathcal{K}| = 2^{H(K)}$ chiavi possibili equiprobabili, il valore che ci si aspetta per il numero di decrittazioni significative *usando tutte le chiavi* è pari a

$$2^{H(K)-nD}$$

Se $H(K) >> nD_n$ ci sono molte decrittazioni significative e di conseguenza è bassa la probabilità di risalire alla chiave corretta che porta al messaggio in chiaro. Se viceversa $H(K) << nD_n$ non c'è incertezza residua sulla chiave. Risulta allora comodo introdurre la quantità

$$n_0 = \frac{H(K)}{D} \tag{7.40}$$

che si definisce *distanza di unicità*. Essa rappresenta la lunghezza di crittogramma (mediamente) necessaria e sufficiente per ottenere un'unica decrittazione significativa.

Se tutti i messaggi sono significativi $R = R_0$, la ridondanza è nulla e la distanza di unicità tende all'infinito. Tale circostanza si giustifica euristicamente dicendo che in tal caso non possiamo attuare la procedura che consiste nell'eliminare le chiavi che portano a messaggi non significativi, giacché essi non esistono. In tal modo si può ottenere una segretezza perfetta anche con una chiave di dimensione finita. Come esempio si pensi a una successione di cifre in base 10 che rappresentino i saldi nei conti correnti di un'insieme di clienti di una banca. In questo caso non esistono sequenze "non significative", poiché qualunque successione è ugualmente verosimile. Anche un cifrario a rotazione è sufficiente per ottenere una segretezza perfetta.

Oss. 7.3. Un metodo per rendere significative tutte le sequenze è quello di attuare una *compressione dei dati*; in tal modo la ridondanza viene rimossa e la distribuzione di probabilità delle lettere secondarie diventa sostanzialmente uniforme.

L'espressione (7.40) della distanza di unicità è valida solo quando n è molto grande, cioè quando la ridondanza D è sostanzialmente indipendente da n. Quando si applica la formula nei casi concreti bisogna fare attenzione alla validità di quest'ipotesi.

Esempio 7.4. Facciamo alcuni esempi relativi ai cifrari classici.

Cifrario a rotazione. Usiamo sempre il valore $||\mathcal{A}|| = 21$ per la cardinalità dello alfabeto. Nel cifrario di rotazione ci sono 21 chiavi, e dunque $H(K) = \log 21 \approx 4.4$. Prendendo il valore asintotico $D_\infty = 3.1$ per la ridondanza si ottiene una distanza di unicità pari a

$$n_0 = \frac{H(K)}{D} = \frac{4.4}{3.1} = 1.42$$

che è un valore molto basso, per il quale la corrispondente ridondanza ha un valore molto diverso da D_∞. Se con l'aiuto della tabella 7.2 usiamo un valore più verosimile, p.es. $D_4 = 1.16$, troviamo $n_0 = \frac{4.4}{1.16} = 3.8$, che è un valore più ragionevole e in linea con i risultati dell'esempio 7.3.

Sostituzione monoalfabetica. Si ha $||\mathcal{K}|| = 21!$ con $H(K) = \log 21! = 65.5$

$$n_0 = \frac{65.5}{3.1} = 21.1$$

In questo caso il risultato è verosimile, essendo adeguato l'impiego di D_∞ per $n \approx 65$.

Cifrario di Vigenère. Il numero delle chiavi dipende dal periodo della parola chiave; si ha infatti $||\mathcal{K}|| = 21^p$ e $H(K) = \log 21^p = 4.392p$ e anche la distanza di unicità manifesta tale dipendenza

$$n_0 = \frac{4.4}{3.1} = 1.42p$$

Naturalmente la formula vale solo per valori rilevanti di p, tali cioè da giustificare l'uso di D_∞.

Capitolo 8
Cifratura a chiave segreta

8.1 Cifrari a blocco

Si è già detto che i cifrari a sostituzione e trasposizione, anche se inutili quando presi individualmente, possono essere concatenati tra di loro e abbinati con altre trasformazioni elementari, in modo da aumentare il livello di sicurezza del sistema complesso così costituito. In ogni caso le trasformazioni concatenate portano da n bit del testo in chiaro a n bit del testo cifrato, e dunque il cifrario composto può essere pensato come se fosse un cifrario a sostituzione semplice che lavora su blocchi di n-ple piuttosto che su singoli bit. Il parametro n è detto *lunghezza del blocco* e ha spesso un ruolo rilevante nella sicurezza del cifrario, poiché è legato in qualche misura alla lunghezza della chiave (si ricordi a tal riguardo il teorema 7.4).

Tuttavia il ricorso a cifrature concatenate e l'aumento della complessità del sistema non sono sempre garanzia di una sicurezza maggiore, che deve comunque essere verificabile direttamente o indirettamente. In qualche caso tale aumento determina anzi una pericolosa involuzione che porta a sistemi complessi più deboli dei sottosistemi costituenti. Da questo punto di vista i sistemi a blocco sono in un certo senso antitetici ai cifrari a flusso. In questi ultimi, infatti, la verifica della sicurezza è attestabile sulla base di un unico carattere del sistema complessivo, che nel caso dei cifrari a flusso è grosso modo assimilabile alla pseudocasualità delle sequenze che escono dal generatore (anche se la pseudocasualità impone la verifica di numerosi test). Di conseguenza si sa chiaramente su quali parametri agire per (tentare di) migliorare significativamente le prestazioni del cifrario o anche per attaccarlo. Anche per i cifrari a chiave pubblica succede qualcosa di analogo, poiché in questo caso il cifrario è associato a un qualche problema computazionale specifico, e tutta la sicurezza (o l'eventuale debolezza) si giocano in questo contesto.

Per i cifrari a blocco la situazione è un po' diversa, poiché la sicurezza non è localizzata, ma deriva da un'interazione sinergica dei diversi blocchi elementari, singolarmente deboli, che costituiscono il cifrario. Ciò comporta che tanto

la fase creativa che la fase di crittanalisi di un buon cifrario a blocco imponga numerosi controlli su tutti gli elementi costitutivi.

In ogni caso la ricetta seguita dai progettisti è quella di dosare sapientemente *diffusione* e *confusione* (si veda 7.3.3), cioè gli ingredienti base suggeriti nell'analisi che Shannon fece della cifratura a blocco. Ciò richiede il soddisfacimento dei seguenti

Criteri empirici per un cifrario a blocco

- ciascun bit del crittogramma dovrebbe dipendere da tutti i bit della chiave e tutti i bit del messaggio;
- non dovrebbero esserci relazioni statistiche evidenti tra messaggio e crittogramma;
- l'alterazione di un bit qualsiasi del messaggio o della chiave dovrebbe alterare ciascun bit del crittogramma con probabilità 1/2; viceversa l'alterazione di un bit del crittogramma dovrebbe portare a un'alterazione imprevedibile dei bit del messaggio.

In questa sezione descriveremo brevemente due sistemi, *DES* e *IDEA*, che sono piuttosto simili per filosofia generale. Il *DES*, suggerito come standard nel 1977 dal *National Bureau of Standards* (*NBS*, ente statunitense del *Department of Commerce*), ha una credibilità molto rilevante, conquistata grazie a due fattori decisivi:

1. essendo molto più vecchio ha avuto modo di subire attacchi molto massicci, che ne hanno accresciuto la rispettabilità; tali attacchi non hanno infatti messo in luce debolezze strutturali del cifrario (è un chiaro esempio di sicurezza attestata indirettamente);
2. il *DES* è stato impiegato come standard dal governo degli USA.

Entrambi i sistemi si basano sull'impiego iterato di trasformazioni per sostituzione e per trasposizione. Inoltre il blocco di testo in chiaro, di lunghezza $n = 64$ in entrambi i casi, viene successivamente scomposto in sottoblocchi (2 da 32 bit per il *DES* e 4 da 16 bit per *IDEA*) che vengono rimescolati da destra a sinistra e viceversa ad ogni iterazione.

8.1.1 Il Data Encryption Standard

Il *DES* venne concepito nei primi anni '70 dai ricercatori della IBM, a seguito di un sollecito del *NBS*, e venne presentato ufficialmente nel 1975. Dopo due anni e numerose discussioni sulla sua sicurezza, nonostante alcune accuse di inadeguatezza mosse da autorevolissimi esperti del settore (W. Diffie e M.E.

Hellman, gli inventori della crittografia a chiave pubblica!) esso venne accolto come standard nel 1977. Il *DES* viene usato correntemente per la protezione di informazioni non classificate e per la gestione di transazioni bancarie, anche se il *NBS* rinnova lo stato di standard mediante controlli periodici (ogni 5 anni) che attestano l'adeguatezza del sistema.

Come ben si osserva su [75], la caratteristica rivoluzionaria di questo cifrario, da un punto di vista per così dire sociologico, è che il suo funzionamento venne specificato in ogni minimo dettaglio proprio nel momento in cui doveva essere lanciato come standard, rinunciando a priori ai vantaggi, che spesso sono solo illusori, che derivano in genere dalla secretazione dell'algoritmo di funzionamento di un cifrario. Ciò riflette una totale adesione al principio di *Kerckhoffs*, secondo il quale bisogna affidare tutta la segretezza del sistema *solo* alla chiave. Inoltre l'aver presentato il cifrario in tutti i suoi dettagli tecnici consente effettivamente di mantenere aperta una sfida, a tempo indeterminato, fra crittografo e crittanalista: gli insuccessi di quest'ultimo sono in fondo la miglior garanzia di sicurezza che uno standard possa offrire. Pochi infatti si fiderebbero di uno standard che nessuno, al di là del costruttore, abbia mai studiato in modo approfondito.

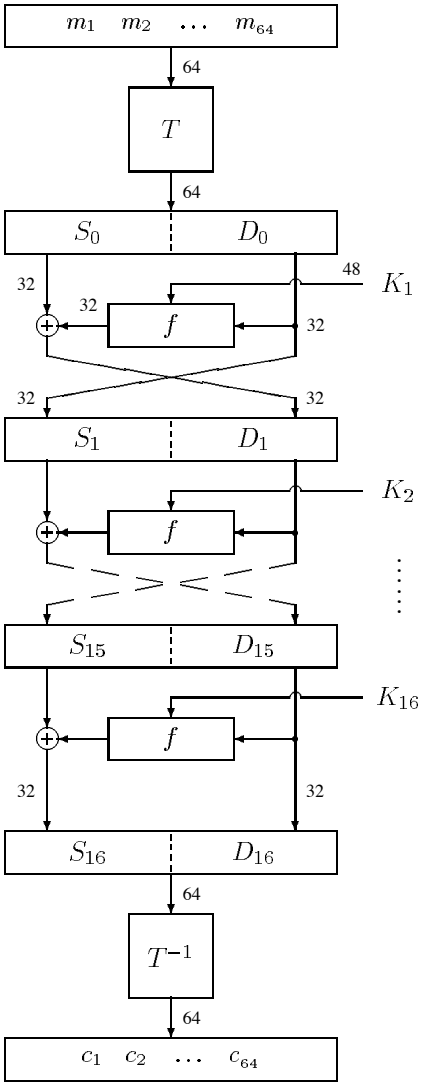
In passato si è anche pensato che l'ente costruttore (o l'ente gestore dello standard), disponesse di qualche "chiave" speciale per poter accedere facilmente alle informazioni trattate dal *DES*, ma di ciò non si è mai avuto riscontro oggettivo.

Passiamo alla descrizione dell'algoritmo, che è lo stesso tanto per la cifratura che per la decifrazione. Il flusso dei messaggi viene segmentato in blocchi da 64 bit; ogni blocco rappresenta l'ingresso dell'algoritmo, che viene alimentato anche dai 56 bit della chiave. La struttura generale, riportata in figura 8.1, è di tipo iterativo: detti S_i e D_i i due semiblocchi sinistro e destro, la struttura di base della singola iterazione è del tipo

$$S_i = D_{i-1} \qquad D_i = S_{i-1} \oplus f(D_{i-1}, K_i) \tag{8.1}$$

dove K_i è una *chiave parziale* di 48 bit che viene ricavata dalla chiave generale del sistema, e $f(.,.)$ è una funzione opportuna. La (8.1) è nota in letteratura col nome di *iterazione di Feistel* .

Il blocco in chiaro d'ingresso, di 64 bit, viene affidato a una matrice di trasposizione T; i bit in uscita vengono suddivisi nelle componenti sinistra e destra (S_i e D_i), da 32 bit ciascuna. Successivamente intervengono le 15 iterazioni descritte dalla (8.1), più un'ultima iterazione in cui S_{16} e D_{16} non vengono più scambiati tra loro, cosicché $S_{16} = S_{15} \oplus f(D_{15}, K_{16})$ e $D_{16} = D_{15}$. I 64 bit di uscita $D_{16}S_{16}$ vengono infine trasposti da una matrice T^{-1} che è l'inversa della T iniziale. La struttura della funzione $f(D_{i-1}, K_i)$ è riportata in figura 8.2. Essa accetta in ingresso i 32 bit del blocco D_{i-1}, e mediante un'espansione



La trasposizione iniziale T

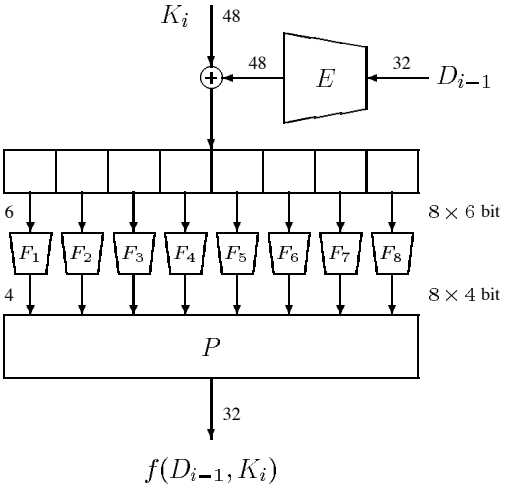
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

La trasposizione finale T^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Figura 8.1 Struttura dell'algoritmo *DES*.

attuata dalla matrice E li trasforma in un blocco di 48 bit, che vanno sommati mod 2 con i 48 bit della chiave parziale K_i, che si ricava in modo opportuno dalla chiave generale di 56 bit. Il blocco risultante viene suddiviso in 8 blocchetti da 6 bit, i quali vengono applicati ad altrettante funzioni di sostituzione F_i, che costituiscono il vero e proprio nucleo non lineare del sistema (si veda figura 8.3). La sostituzione avviene nel modo seguente: se $b_1b_2b_3b_4b_5b_6$ è il blocco



L'espansione *E*

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

La trasposizione *P*

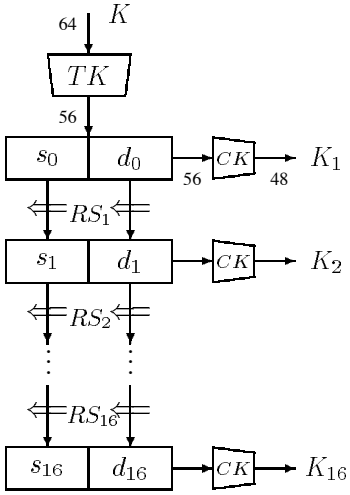
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Figura 8.2 La funzione $f(D_{i-1}, K_i)$.

d'ingresso alla F_i esso viene suddiviso nei due sottoblocchi b_1b_6 e $b_2b_3b_4b_5$, che vengono letti come rappresentazione binaria degli interi $0 \leq r \leq 3$ e $0 \leq c \leq 15$ rispettivamente. r e c determinano la riga e la colonna della tabella 8.3 dall'incrocio delle quali si ricava l'intero u, la cui rappresentazione binaria $u_1u_2u_3u_4$ diventa il blocchetto in uscita di lunghezza 4. Gli 8 blocchetti di lunghezza 4 vengono poi fatti passare attraverso la trasposizione P, la cui uscita costituisce la $f(D_{i-1}, K_i)$. Il calcolo delle chiavi parziali avviene invece nel modo seguente. Si parte dalla chiave globale, costituita da 8 byte di cui l'ultimo usato come controllo di disparità. Il numero di bit effettivi per la chiave è allora 56, con i quali è possibile costruire $2^{56} \approx 7.20\ 10^{16}$ chiavi diverse. Il blocco di 64 bit viene immesso nella matrice di trasposizione TK (si veda figura 8.4); quest'ultima prende in considerazione solamente i 56 bit effettivi della chiave. L'uscita viene suddivisa in due sottoblocchi s_i e d_i da 28 bit ciascuno, sui quali viene iterata una rotazione verso sinistra RS di ampiezza 1 o 2, a seconda del livello dell'iterazione, secondo la tabella sotto riportata

b_1b_6 \ $b_2b_3b_4b_5$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	F_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	F_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	F_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	F_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	F_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	F_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	F_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	F_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Figura 8.3 La tabella delle sostituzioni F_i.



La trasposizione *TK*

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

La compressione *CK*

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Figura 8.4 Calcolo delle chiavi parziali per il *DES*.

Iterazione	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RS	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Per rotazione a sinistra s'intende una traslazione a sinistra dei bit con riporto all'ingresso dei bit in eccesso. Se dunque $\beta_1\beta_2\ldots\beta_{28}$ è un blocchetto sottoposto a una RS pari a 2, dopo la rotazione si ottiene $\beta_3\beta_4\ldots\beta_1\beta_2$. La decifrazione avviene alimentando lo stesso algoritmo con il crittogramma, ma usando le chiavi parziali nel senso inverso, cioè $K_{16}, K_{15}, \ldots, K_1$.

Si è dibattuto a lungo sulla sicurezza offerta dal sistema *DES*, ipotizzando anche degli elaboratori dedicati specificamente alla forzatura bruta del sistema mediante una ricerca su tutte le possibili chiavi (che è esattamente lo scopo che ogni progettista di cifrari si prefigge!). In qualche caso è stata data persino una stima del costo di una tale macchina. La sostanza è che il *DES* ha svolto in modo impeccabile il suo ruolo, poiché a distanza di 25 anni esso mantiene inalterata la sua rispettabilità, anche se a partire dagli anni '90 sono stati attuati diversi attacchi crittanalitici, basati sui concetti di crittanalisi *lineare e differenziale*, che hanno consentito di abbassare la soglia di complessità a valori dell'ordine di 2^{43} (sia pure facendo un uso massiccio del testo in chiaro)[61]. L'unica critica consistente, e che ha decretato il recente abbandono dello status di standard, deriva dalla grandezza della chiave (56 bit), giudicata non più adeguata ai

tempi. Altri elementi degni di nota sono la presenza di 4 *chiavi deboli*, per le quali $DES_K(DES_K(m)) = m,\ \forall m$ e di 6 *chiavi semideboli*, che portano a $DES_{K_1}(DES_{K_2}(m)) = m$.

8.1.2 Il sistema *IDEA*

Diamo ora un breve cenno sul cifrario *IDEA* (International Data Encryption Algorithm), proposto recentemente come possibile sostituto del *DES*. Esso impiega una generalizzazione dell'iterazione di Feistel , nel senso che il blocco iniziale di 64 bit del messaggio in chiaro viene suddiviso in 4 sottoblocchi M_1, M_2, M_3, M_4 da 16 bit (invece che in 2 da 32 come nel *DES*) che vengono fatti interagire con 6 chiavi parziali $K_i^{(j)}$ $(1 \leq i \leq 6)$ da 16 bit, ricavate a partire dalla chiave generale di 128 bit. Successivamente vengono eseguite 8 iterazioni del tipo di quelle riportate in figura 8.5 $(1 \leq j \leq 6)$, e a ogni iterazione il gruppo di sei chiavi parziali viene cambiato. Alla fine delle iterazioni il risultato viene nuovamente composto con un ultimo blocco di 4 chiavi, $K_1^{(9)}, K_2^{(9)}, K_3^{(9)}, K_4^{(9)}$ similmente a quanto accade nella parte alta del disegno. L'aspetto più interessante del cifrario è dato dal fatto che vengono usate in modo sapiente operazioni di tipo diverso sulle diverse strutture di gruppo associate all'insieme delle 2^n n-ple binarie, cioè la somma bit per bit mod 2, la somma mod 2^{16} e il prodotto mod $2^{16}+1$ (con lo zero interpretato come 2^{16}), che dovrebbero portare a un cifrario a blocco che soddisfa i criteri empirici riferiti nella sezione 8.1. Il cifrario è stato oggetto recentemente di numerosi attacchi, che hanno avuto successo su versioni ridotte dello stesso (ridotto numero di iterazioni) [14].

8.2 Cifrari a flusso e la generazione di sequenze pseudocasuali

Il sistema *one-time pad*, analizzato nel paragrafo 7.4.2, introduce alla classe dei *cifrari a flusso*, nei quali al flusso dei messaggi (p.es. una sequenza binaria ricavata dall'uscita di un codificatore di sorgente) viene sovrapposto il flusso delle chiavi, che nel caso del *one-time pad* è costituito da una sequenza aleatoria, stazionaria, senza memoria e con una d.p. uniforme. Usando in ricezione la stessa sequenza come chiave di decifrazione si può ricavare il messaggio in chiaro (si veda figura 7.6).

Dal punto di vista operativo la difficoltà di attuazione non sta tanto nella costruzione di una vera e propria sorgente aleatoria, poiché molti fenomeni fisici offrono la possibilità di ottenere delle ottime sequenze aleatorie. Come esempio immediato si pensi alla tensione di rumore di una resistenza alimentata da una tensione continua e operante a temperatura ambiente; l'oscillazione aleatoria attorno al valore medio (corrispondente alla tensione continua nominale di alimentazione) può essere facilmente campionata a frequenza costante, attribuendo

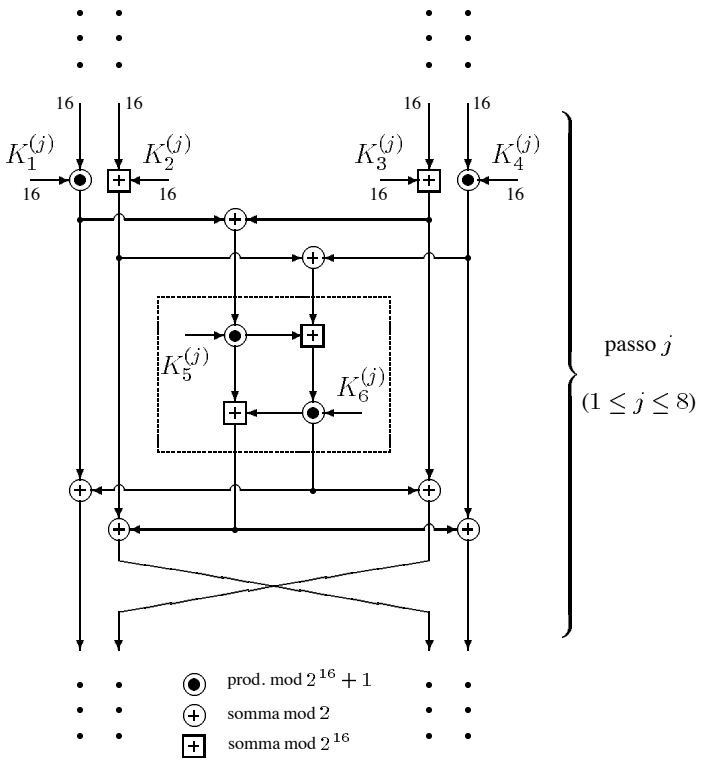


Figura 8.5 Struttura dell'algoritmo *IDEA*.

p.es. "0" a un campione sotto valor medio e "1" a un campione sopra il valore medio.

Il vero problema da superare è quello della chiave, che dovendo essere trasmessa a destinazione rende il sistema di difficilissima gestione. D'altra parte le prestazioni di perfezione e idealità del sistema sono molto allettanti, e di conseguenza risulta ragionevole tentare una semplificazione dello schema per guadagnare qualcosa sul piano operativo, rinunciando beninteso alla perfezione teorica.

La semplificazione introdotta è quella di sostituire la sequenza aleatoria con una sequenza generata in modo *deterministico*, sì da poter usare lo stesso algoritmo di generazione tanto in trasmissione che in ricezione. La struttura effettiva della sequenza che esce dalla sorgente, che chiamiamo *pseudoaleatoria* o *pseudocasuale*, è determinata dalla chiave d'innesco dell'algoritmo, cioè dalla informazione che usiamo per generare sequenze diverse a partire dallo stesso algoritmo. È solo quest'ultima la vera e propria chiave di cifratura/decifrazione,

che in quanto informazione d'innesco di un determinismo ha un ingombro molto contenuto ed è dunque gestibile secondo la prassi ordinaria d'impiego di un canale sicuro e ad alto costo.

8.2.1 Caratteri delle sequenze aleatorie

Il problema della realizzazione di un cifrario a flusso adeguato si riduce dunque a quello della *costruzione di un buon generatore di sequenze pseudocasuali*, che sarà di norma costituito da una opportuna *macchina sequenziale a stati finiti* che usa la chiave di cifratura come stato iniziale; dalla successione (periodica) degli stati si otterrà una sequenza da sovrapporre al flusso dei messaggi. Affinché il sistema si comporti con buona approssimazione *come se fosse* un sistema *one-time pad*, è necessario che la sequenza pseudocasuale abbia impressi in sé i caratteri distintivi delle sequenze aleatorie, in modo tale che l'eventuale oppositore non riesca a distinguere la *sequenza pseudocasuale* generata deterministicamente da una sequenza generata in modo squisitamente aleatorio.

Il generatore deve pertanto possedere i seguenti requisiti:

G1 generare sequenze pseudocasuali;

G2 essere dotato di un periodo grande a piacimento;

G3 essere resistente agli attacchi crittanalitici con testo in chiaro.

Si noti che **G2** è in qualche misura ricompresa nella **G3**, in quanto un attacco con testo in chiaro con un periodo della chiave non adeguato consentirebbe di ricavare l'intero periodo e dunque forzare immediatamente il cifrario. La **G3** è un punto molto delicato, poiché come vedremo la validità della **G1** non è di per sé sufficiente a garantire la sicurezza del sistema. Per quel che attiene **G1** si tratta invece di fornire opportuni *criteri* di pseudocasualità, per decidere se una sequenza assegnata possa o meno definirsi pseudocasuale. Dobbiamo allora individuare i *caratteri* tipici delle sequenze squisitamente aleatorie, cercando di ottenere con le sequenze deterministiche un comportamento analogo. La conoscenza di questi caratteri consente inoltre di elaborare dei *test statistici* opportuni, che verranno successivamente applicati alle sequenze generate deterministicamente, decretando o meno la loro ammissibilità in ambito crittografico.

Analizziamo dunque la struttura delle sequenze che escono da una sorgente $\mathcal{S}$ aleatoria, discreta, binaria, stazionaria, senza memoria e con d.p. uniforme, che rappresentiamo simbolicamente con la notazione

$$\mathcal{S} = \begin{pmatrix} 0 & 1 \\ 1/2 & 1/2 \end{pmatrix} \tag{8.2}$$

Il comportamento delle sequenze aleatorie può essere analizzato usando il teorema di Bernoulli 3.4 e la (3.19), con riferimento ai caratteri di seguito elencati.

Frequenza di 0 e 1. Poiché la probabilità a priori dell'uscita di 0 e 1 è $1/2$, sulla base della (3.19) ci aspettiamo che la frequenza relativa di entrambi tenda a 1/2 in probabilità quando la lunghezza n della sequenza tende all'infinito; dunque, seguendo la (3.11) dev'essere

$$\lim_{n\to\infty} \text{ prob } \frac{n_0}{n} = \lim_{n\to\infty} \text{ prob } \frac{n_0}{n} = 1/2 \tag{8.3}$$

dove n_0, n_1 è il numero di 0 (1) della sequenza. In altre parole, presa una porzione sufficientemente grande di sequenza, deve verificarsi l'uguaglianza approssimata $n_0 \cong n_1 \cong 1/2$, cioè il numero di zeri è circa uguale al numero degli uni (in probabilità).

Frequenza delle tratte. Il fatto che in una sequenza binaria a d.p. uniforme ci siano circa metà zeri e circa metà uni non è però sufficiente a garantire che essi siano correttamente *distribuiti*. Per esempio la sequenza $0\ldots01\ldots1$ costituita dalla prima metà di zeri e dalla seconda di uni soddisfa senz'altro il requisito di prima, ma non rassomiglia sicuramente a una *tipica* sequenza aleatoria. Per tener conto del rimescolamento tra zeri e uni è necessario introdurre le *tratte* (unitarie o nulle), cioè delle sottosuccessioni di uni (zeri) precedute e seguite da uno zero (uno). La 01110 è allora una tratta unitaria di lunghezza 3, mentre la 100001 è una tratta nulla di lunghezza 5, e così via. La distribuzione tipica delle tratte che troviamo su una sequenza aleatoria può essere dedotta facilmente analizzando tutte le possibili sequenze di lunghezza n (che sono equiprobabili) e contando le tratte. Per $n = 2, 3, 4$ si ha lo schema sotto riportato

$$\begin{array}{lll}
 & & \left.\begin{array}{l}\mathbf{0000}\\ \mathbf{1111}\end{array}\right\} \geq 4 \\
 & & \left.\begin{array}{l}\mathbf{000}1\\ \mathbf{111}0\end{array}\right\} = 3 \\
 & \left.\begin{array}{l}\mathbf{000}\\ \mathbf{111}\end{array}\right\} \geq 3 & \left.\begin{array}{l}\mathbf{00}10\\ \mathbf{00}11\\ \mathbf{11}01\\ \mathbf{11}00\end{array}\right\} = 2 \\
\left.\begin{array}{l}\mathbf{00}\\ \mathbf{11}\end{array}\right\} \geq 2 & \left.\begin{array}{l}\mathbf{00}1\\ \mathbf{11}0\end{array}\right\} = 2 & \\
\left.\begin{array}{l}\mathbf{0}1\\ \mathbf{1}0\end{array}\right\} = 1 & \left.\begin{array}{l}\mathbf{0}11\\ \mathbf{0}10\\ \mathbf{1}00\\ \mathbf{1}01\end{array}\right\} = 1 & \left.\begin{array}{l}\mathbf{0}100\\ \mathbf{0}101\\ \mathbf{0}110\\ \mathbf{0}111\\ \mathbf{1}000\\ \mathbf{1}001\\ \mathbf{1}010\\ \mathbf{1}011\end{array}\right\} = 1
\end{array}$$

nel quale sono evidenziate le tratte (in grassetto) e la lunghezza delle stesse. Si noti che le tratte sono tutte iniziali, metà sono nulle e metà unitarie,

e di conseguenza ogniqualvolta inizia una tratta essa è nulla o unitaria con probabilità 1/2. Inoltre una tratta di lunghezza j (senza distinguere tra nulla e unitaria) ha probabilità a priori pari a

$$\Pr(T(j)) = \frac{1}{2^j} \tag{8.4}$$

La sorgente (8.2) può allora essere considerata equivalente alla seguente *sorgente (infinita) delle tratte*

$$\mathfrak{S}_T = \begin{pmatrix} 0 & 1 & 00 & 11 & 000 & \dots & 00\dots0 & 11\dots1 & \dots \\ 1/4 & 1/4 & 1/8 & 1/8 & 1/16 & \dots & 1/2^{j+1} & 1/2^{j+1} & \dots \end{pmatrix}$$

che genera una tratta (unitaria o nulla) di lunghezza j con probabilità pari a $1/2^{j+1}$; essa ha esattamente lo stesso comportamento della (8.2). Detto T_j il numero di tratte di lunghezza j e T il numero totale di tratte su una certa sequenza di lunghezza n, l'applicazione della legge dei grandi numeri porta a

$$\lim_{n\to\infty} \text{prob}\, \frac{T_j}{T} = \frac{1}{2^j} \tag{8.5}$$

Il discorso fatto per le tratte può essere ripetuto anche per le k-ple, considerando al posto della (8.2) la sua *estensione* k-esima. Ciò porta alla seguente *sorgente delle n-ple*

$$\mathfrak{S}_k = \begin{pmatrix} 00\dots00 & 00\dots01 & 00\dots10 & \dots & 11\dots10 & 11\dots11 \\ 1/2^k & 1/2^k & 1/2^k & \dots & 1/2^k & 1/2^k \end{pmatrix}$$

e al corrispondente limite in probabilità

$$\lim_{n\to\infty} \text{prob}\, \frac{N_j}{N} = \frac{1}{2^k} \tag{8.6}$$

dove N_j e il numero di k-ple di tipo j sulle N k-ple complessive della sequenza di lunghezza n.

Assenza di memoria. La presenza di fenomeni di memoria associati a una sequenza preassegnata di lunghezza n può essere messa in luce mediante la *funzione di autocorrelazione*,

$$C(t) = \frac{\#\text{ coinc.} - \#\text{ discor.}}{n} = \frac{s_0 - s_1}{n} = \frac{n - 2s_1}{n} \tag{8.7}$$

che è in grado d'individuare eventuali correlazioni tra l'uscita all'istante i e l'uscita a un certo istante $i - t$ ($t > 0$). Si tratta essenzialmente di confrontare la sequenza con la sua traslata di t posizioni, rilevando le coincidenze e le differenze (nel caso binario) che si rilevano tra le due sequenze.

Poiché una coincidenza corrisponde a uno "0" nella sequenza somma (bit per bit mod 2), mentre una discordanza corrisponde a "1", ci si può basare direttamente sul numero s_0 di 0 e s_1 di 1 della somma. Un'espressione equivalente è la

$$C(t) = 1 - \frac{4}{n} n_1 + \frac{4}{n} n_{11}$$

che deriva dalla computazione del numero n_1 di "1" della sequenza di partenza e del numero n_{11} di coincidenze tra "1" che si realizzano tra la sequenza e la sua traslata. Per ricavare quest'ultima espressione dalla (8.7) basta tener conto che i contributi che portano a un 1 della sequenza somma sono del tipo $0 \oplus 1$ oppure $1 \oplus 0$. I primi sono in numero di $n_1 - n_{11}$ e lo stesso vale per i secondi, dove n_1 è a rigore il numero di uni della sequenza traslata. Dunque $s_1 = 2(n_1 - n_{11})$. Naturalmente $C(0) = 1$, poiché tra una sequenza e sé stessa ci sono solo coincidenze.

Per esempio la sequenza 011010011, per la quale si illustrano le prime tre traslazioni

	$t = 1$	$t = 2$	$t = 3$
sequenza	011010011	011010011	011010011
seq. traslata	110100110	101001101	010011011
somma	101110101	110011110	001001000

è caratterizzata dai seguenti valori

C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)
-1/3	-1/3	5/9	-1/3	-1/3	5/9	-1/3	-1/3

Se una sequenza è aleatoria il numero degli zero della sequenza somma è circa uguale al numero degli uno della stessa sequenza, e dunque $C(t) \approx 0$. Se viceversa si verifica una forte correlazione per una certa traslazione t, la funzione di autocorrelazione in quel punto assume un valore significativamente diverso dallo zero (positivo o negativo). Nel caso dell'esempio si nota che per $t = 3$ e $t = 6$ la $C(r)$ assume un valore elevato (positivo), e dunque c'è una forte similarità fra la sequenza e la sua traslata di 3 e 6 posizioni, evidenziato dal fatto che sequenza e traslata sono per questi valori di t pressoché uguali (o complementari se il valore è negativo). Ciò significa che avendo una porzione della sequenza di partenza (derivante p.es. da un attacco con testo in chiaro) si possono fare delle previsioni sulla struttura della stessa sequenza a partire dalla posizione t con una probabilità di successo molto elevata.

Riassumendo, per una sequenza aleatoria vale il seguente limite in probabilità

$$\lim_{n\to\infty} \text{prob } C(t) = 0 \qquad \forall t\ (t \neq 0) \tag{8.8}$$

I tre caratteri messi in luce, numero degli 0 e 1, frequenza delle tratte (o delle n-ple) e funzione di autocorrelazione, non sono necessariamente i soli che si possono prendere in considerazione, ma sono quantomeno i più immediati (si noti peraltro che non sono neanche indipendenti, poiché il secondo comprende anche il primo). Ci sono numerose altre caratteristiche statistiche che possono essere impiegate con successo per specificare in modo sempre più accurato il comportamento delle sequenze aleatorie pure; per una trattazione più completa del problema rimandiamo senz'altro a [49], e anche a [5] e alle relative bibliografie. In ogni caso, dal punto di vista pratico si tratta di decidere in che senso e con quale livello di approssimazione far riferimento ai limiti espressi dalle (8.3), (8.5) e (8.8).

Storicamente il primo tentativo di definire delle condizioni per decretare la pseudocasualità di una sequenza fu compiuto da Golomb [40]; in tal caso i tre limiti in probabilità sopra citati vengono sostituiti con delle crude uguaglianze, che portano ai cosiddetti

Criteri di pseudocasualità di Golomb.

Gol 1. Il numero degli zeri uguaglia il numero degli uni (a meno di un'unità).

Gol 2. Ci sono metà tratte di lunghezza 1, un quarto di tratte di lunghezza 2, un ottavo di tratte di lunghezza 3, ... ,$1/2^j$ tratte di lunghezza j, e per ogni insieme di tratte di pari lunghezza, metà sono nulle e metà unitarie (a meno di un'unità).

Gol 3. La funzione di autocorrelazione $C(t)$ è costante (se $t \geq 1$).

S'intuisce che la validità dei criteri impone vincoli molto stringenti, giacché vengono escluse moltissime sequenze che, pur non soddisfacendoli pienamente, risultano senz'altro più che adeguate agli impieghi crittologici; in un certo senso detta validità finisce addirittura con l'identificare univocamente le sequenze che li soddisfano, che sono molto poche. Esse sono note col termine di *pn-sequenze* (da *pseudonoise sequences*), e sono costituite dall'insieme di tutte (e sole) le sequenze generate da un registro lineare a scorrimento retroazionato associato a un polinomio primitivo. Si noti però che i criteri di Golomb furono introdotti in un contesto alquanto diverso dalla crittografia, e che solo successivamente vennero ripresi come punto di partenza per lo studio della pseudocasualità anche in in ambito crittologico.

Si può ottenere una versione più adeguata dei criteri ammorbidendo l'uguaglianza in senso stretto con un'uguaglianza al limite, quando $n \to \infty$, in modo da ottenere i seguenti

Criteri estesi di Golomb

Est 1. $\lim_{n\to\infty} n_0 = \lim_{n\to\infty} n_1 = \frac{1}{2}$

Est 2. $\lim_{n\to\infty} \frac{T_j}{T} = \frac{1}{2^j}$

Est 3. $\lim_{n\to\infty} C(t) = 0 \qquad (t > 0)$

Da questo punto di vista definiamo psudocasuale una sequenza periodica che soddisfa i tre criteri estesi quando la lunghezza n del periodo tende all'infinito.

Si tenga però conto che in pratica si ha la necessità di valutare una singola sequenza preassegnata, più che un'intera famiglia di sequenze con un comportamento asintotico uniforme. Per questo motivo si ricorre il più delle volte a dei test statistici, che fanno riferimento alle caratteristiche viste prima (ed eventualmente ad altre), in modo tale da decretare la pseudocasualità della singola sequenza in esame nel caso in cui questa superi tutti i test prescritti. Un test molto interessante, che qua accenniamo solamente rimandando il lettore a [62] per i dettagli, è il *test universale di Maurer*; esso prende spunto dall'osservazione che una sequenza generata da una sorgente binaria, aleatoria, stazionaria, senza memoria e con d.p. uniforme non è (mediamente) comprimibile. Se dunque una certa sequenza può essere compressa in modo significativo bisogna dichiararla non pseudocasuale. Tutto ciò è legato in modo rilevante alla *complessità lineare* delle sequenze che tratteremo più avanti nel paragrafo 8.2.3 (e in ultima analisi alla complessità di Kolmogorov [19]).

La progettazione di un buon generatore di sequenze pseudocasuali richiede dunque la verifica a priori della qualità delle sequenze generate; detta verifica può essere fornita in qualche caso fortunato per via diretta, *dimostrando* che la famiglia delle sequenze soddisfa i criteri di pseudocasualità estesi, oppure per via indiretta, sottoponendo le sequenze in uscita a un certo insieme di test statistici.

Resta aperto il problema di *come* generare sequenze pseudocasuali per impieghi crittologici, soprattutto tenendo conto della resistenza agli attacchi crittanalitici con testo in chiaro che tali sequenze, a norma della **G3** vista precedentemente, devono possedere. Poiché la maggior parte dei generatori è basata sull'impiego dei registri a scorrimento retroazionato, analizzeremo dettagliatamente le prestazioni di interesse crittografico di questi dispositivi, già introdotti e usati proficuamente nell'ambito dei codici correttori d'errore.

8.2.2 I registri a scorrimento retroazionato

Tra le macchine sequenziali a stati finiti di cui si è fatto cenno precedentemente vale la pena di analizzare i registri a scorrimento retroazionato (*RLSR*), introdotti nella sezione 6.5 con lo scopo di effettuare codifiche e decodifiche cablate per i codici correttori. In quel caso si è posto l'accento sulla potenzialità dei registri nell'ambito della computazione sui campi finiti; in quest'ambito ci concentreremo invece sulle sequenze che i registri sono in grado di generare, sulla loro periodicità e sulle caratteristiche di pseudocasualità manifestate dalle stesse.

Consideriamo allora il *RLSR* di tipo 1 di figura 6.3, che riportiamo per comodità in figura 8.6. Ricordiamo che il suo funzionamento è scandito da un

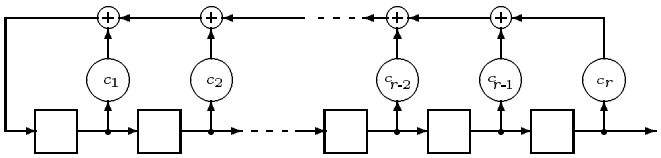


Figura 8.6 *RLSR* di tipo 1 associato al polinomio $1 + c_1x + \ldots + c_rx^r$.

orologio interno; in corrispondenza di ciascun istante di un tempo discreto il contenuto di ciascuna cella viene traslato nella cella successiva, il contenuto dell'ultima esce, mentre la prima cella viene alimentata direttamente dalla *rete di retroazione*, che effettua la somma $\mod 2$ del contenuto di *alcune* tra le celle disponibili. Le celle in questione sono quelle che alimentano un moltiplicatore con un coefficiente $c_i = 1$ (nel caso binario la presenza o meno della connessione viene determinata dal valore del coefficiente di retroazione, che vale 1 nel primo caso e 0 nell'altro).

Inneschiamo ora il registro con uno stato iniziale qualunque. Poiché la rete di retroazione è deterministica (a un certo stato del registro corrisponde sempre la stessa uscita della rete di retroazione), la sequenza degli stati percorsi dal registro dev'essere periodica, e lo stesso vale per la successione in uscita che deriva direttamente dagli stati. Ma gli stati possibili sono 2^r, e questa è anche una limitazione superiore per il periodo. Tenuto poi conto che il raggiungimento dello stato nullo da parte del registro (in cui tutte le celle hanno 0 come contenuto) comporta la permanenza perpetua dello stesso registro in questo stato, si può dedurre la seguente

Proposizione 8.1. *Il periodo p di un registro lineare a scorrimento retroazionato è limitato superiormente da $2^r - 1$.*

Partiamo ora dallo stato iniziale $a_{-1}, a_{-2}, \ldots, a_{-r}$; l'uscita della rete di retroazione negli istanti successivi vale

$$a_0 = \sum_{i=1}^{r} c_i a_{-i} \qquad a_1 = \sum_{i=1}^{r} c_i a_{1-i} \qquad \ldots \qquad a_n = \sum_{i=1}^{r} c_i a_{n-i} \tag{8.9}$$

che è *l'equazione di ricorrenza lineare* del registro (la sommatoria è fatta modulo 2).

Per studiare la struttura e la periodicità delle sequenze in uscita dall'ultima cella (o da qualunque altra, visto che la sequenza che si ottiene è la stessa a meno di traslazioni) è opportuno introdurre la *funzione generatrice*

$$G(x) = \sum_{n=0}^{\infty} a_n x^n \tag{8.10}$$

che è una serie formale costruita con gli elementi della successione. Sostituendo la (8.9) nella (8.10) si ottiene

$$\begin{aligned} G(x) &= \sum_{n=0}^{\infty} \sum_{i=1}^{r} c_i a_{n-i} x^n = \sum_{i=1}^{r} c_i x^i \sum_{n=0}^{\infty} a_{n-i} x^{n-i} = \\ &= \sum_{i=1}^{r} c_i x^i \left(a_{-i} x^{-i} + \ldots + a_1 x^{-1} + G(x) \right) \end{aligned}$$

dalla quale si ricava

$$G(x) = \frac{\sum_{i=1}^{r} c_i x^i \left(a_{-i} x^{-i} + \ldots + a_1 x^{-1} \right)}{1 - \sum_{i=1}^{r} c_i x^i} = \frac{g(x)}{f(x)} \tag{8.11}$$

con $\mathrm{gr}[g(x)] \leq r - 1$ e $\mathrm{gr}[f(x)] = r$. La relazione (8.11) consente di esprimere la successione delle uscite, che è infinita, nei termini dei coefficienti di retroazione c_i e dello stato iniziale $a_{-1}, a_{-2}, \ldots, a_{-r}$, organizzati secondo i polinomi $f(x)$ e $g(x)$. $f(x)$ dipende solo dalla rete di retroazione, e per questo motivo è definito *polinomio caratteristico*. Esso ha un ruolo rilevante nella periodicità della sequenza.

Teorema 8.1. *Se la sequenza (a_n) di un registro di dimensione r si ottiene a partire dallo stato iniziale $a_{-1} = 0, a_{-2} = 0, \ldots, a_{-r+1} = 0, a_{-r} = 1$, allora il periodo della sequenza è il più piccolo intero p tale che $f(x)/1 - x^p$ e viceversa.*

Dim. Se la sequenza ha periodo p, tenendo conto dello stato iniziale scelto

possiamo scrivere la funzione generatrice come

$$\begin{aligned}\frac{1}{f(x)} &= \left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right) + \\ &+ \left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right) x^p + \\ &+ \left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right) x^{2p} + \\ &\vdots \\ &= \left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right)\left(1 + x^p + x^{2p} + \ldots\right) = \\ &= \frac{\left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right)}{1 - x^p} \end{aligned} \tag{8.12}$$

e dunque $f(x)$ divide $1 - x^p$.

Viceversa se $f(x)$ divide $1 - x^p$, essendo

$$f(x)\left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right) = 1 - x^p$$

si giunge subito alla tesi con un procedimento inverso a quello visto prima. □

Si osservi che la validità del teorema sussiste anche partendo da uno stato diverso da $00\ldots01$, purché il polinomio $g(x)$ non abbia fattori in comune con $f(x)$. Se $f(x)$ è *irriducibile*, la lunghezza del periodo *non* dipende da $g(x)$, e quindi dallo stato iniziale. Il valore p si chiama anche $esponente$ di $f(x)$, e se $p = 2^r - 1$ il polinomio si dice *primitivo*.

Teorema 8.2. *Se* (a_n) *ha periodo massimo* $2^r - 1$, *allora* $f(x)$ *è irriducibile*.

Dim. Poiché la sequenza genera tutti i $2^r - 1$ stati prima di ripetersi, essa genera anche lo stato $00\ldots01$. Se prendiamo in considerazione proprio questo stato ci ritroviamo nelle condizioni del teorema precedente; il periodo di (a_n) è allora l'esponente di $f(x)$. Sia per assurdo $f(x) = s(x)t(x)$. Per il teorema di Euclide [33] possiamo scrivere

$$\frac{1}{f(x)} = \frac{h(x)}{s(x)} + \frac{k(x)}{t(x)}$$

con $\text{gr}[s(x)] = r_s > 0$, $\text{gr}[t(x)] = r_t > 0$ $r_s + r_t = r$. Poiché la funzione generatrice è espressa come somma di altre due funzioni generatrici, deve essere

$$\begin{aligned} p = 2^r - 1 &\leq \text{m.c.m.}(2^{r_s} - 1, 2^{r_t} - 1) \leq (2^{r_s} - 1)(2^{r_t} - 1) \leq \\ &\leq (2^{r_s} - 1)\,(2^{r_t} - 1) = 2^r - (2^{r_s} + 2^{r_t}) + 1 \leq 2^r - 3 \end{aligned}$$

che è assurdo. Se infine fosse $f(x) = s(x)^2$ si otterrebbe parimenti

$$p = 2^r - 1 \leq 2\left(2^{r/2} - 1\right) < 2^{r/2}\left(2^{r/2} - 1\right) = 2^r - 2^{r/2}$$

e dunque un altro assurdo. □

Oss. 8.1. Si noti che l'inverso del teorema 8.2 non vale, poiché per esempio il polinomio $x^4 + x^3 + x^2 + x + 1$ è irriducibile (su GF(2)), ma non primitivo; infatti esso divide $1 - x^5$ e le sequenze generate da un registro a esso associato hanno periodo 5.

La lunghezza del periodo è dunque legata alla divisibilità di $f(x)$ per un polinomio nella forma $1 + x^p$ (su GF(2)). Abbiamo già incontrato nella (6.59) e nel teorema 6.8 due metodi per scomporre un polinomio di questo tipo. In particolare la scomposizione nel prodotto dei polinomi ciclotomici (6.59), usata in sinergia con la (6.62), fornisce importanti informazioni sul periodo. Per esempio si trova subito $x^4+x^3+x^2+x+1 = Q^{(5)}(x)$, e dunque $Q^{(5)}(x)$ divide x^5+1, mentre x^4+x^3+1, facendo parte della scomposizione di $Q^{(15)}(x)$, è divisore di $x^{15} + 1$, ed è dunque primitivo avendo periodo $2^4 - 1 = 15$ (si veda l'esempio 6.19).

Dal teorema 6.8, cioè dal fatto che il polinomio $x^{2^r} + x = x\left(x^{2^r-1} + 1\right)$ si scompone nel prodotto di tutti i polinomi monici irriducibili su GF(2) il cui grado divide r, si deduce il seguente

Teorema 8.3. *Se la sequenza (a_n) è associata a un polinomio irriducibile di grado r, il periodo della stessa è un fattore di $2^r - 1$.*

Corollario 8.1. *Se $2^r - 1$ è un numero primo, ogni polinomio di grado r corrisponde a un registro che genera una sequenza di periodo $2^r - 1$.*

Oss. 8.2. I numeri primi nella forma $2^r - 1$, noti col nome di *primi di Mersenne*, costituiscono un affascinante capitolo della teoria dei numeri. Essi sono molto rari e concentrati nei primi valori di r. Al momento ne sono noti solo 33, il più grande dei quali è $2^{859433} - 1$; di questi 33 ben 10 hanno $r < 100$, e sono rispettivamente $2, 3, 5, 7, 13, 17, 19, 31, 61, 89$. Condizione necessaria affinché un numero sia di Mersenne è che r sia primo. Al momento non è noto neanche se essi siano in numero infinito, anche se questa è l'ipotesi più verosimile.

Nella costruzione di un generatore pseudocasuale, basato su *RLSR*, è opportuno avere delle garanzie sulla lunghezza del periodo *a prescindere* dallo stato iniziale scelto. Il teorema 8.1 suggerisce di scegliere una rete di retroazione associata a un polinomio irriducibile. Inoltre, poiché in crittologia siamo interessati a costruire sequenze con un periodo elevato (si veda infatti la caratteristica **G2** vista precedentemente), è anche opportuno scegliere tra tutti gli irriducibili quei polinomi che sono anche primitivi, e che garantiscono dunque il periodo massimo $2^r - 1$. La teoria dei numeri offre delle formule per calcolare il numero di polinomi irriducibili e primitivi per ogni r assegnato; bisogna far nuovamente

riferimento alla *funzione aritmetica di Möbius* $\mu(j)$, definita nella (6.60), e alla *funzione di Eulero* $\Phi(j)$, che esprime il numero di interi minori di j e primi con esso. Una espressione conveniente per $\Phi(j)$ è la seguente

$$\Phi(j) = j\left(1-\frac{1}{j_1}\right)\left(1-\frac{1}{j_2}\right)\ldots\left(1-\frac{1}{j_m}\right) \tag{8.13}$$

dove $j_1, j_2, \ldots, j_m$ sono i fattori primi distinti che entrano nella scomposizione di j. Si può dimostrare (si veda [40]) che il numero $\Psi(r)$ di polinomi irriducibili e $\lambda(r)$ di polinomi primitivi di grado r è dato dalle

$$\Psi(r) = \frac{1}{r}\sum_{d:d/r} 2^d \mu(r/d) \tag{8.14}$$

$$\lambda(r) = \frac{\Phi\left(2^r-1\right)}{r} \tag{8.15}$$

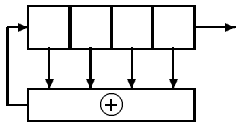
Naturalmente, poiché non tutti i polinomi irriducibili sono primitivi, deve valere anche $\lambda(r) \leq \Psi(r)$.

La tabella 8.1 mostra i primi valori delle due funzioni fino a $r = 10$.

r	2^r-1	$\Psi(r)$	$\lambda(r)$
1	1	2	1
2	**3**	1	1
3	**7**	2	2
4	15	3	2
5	**31**	6	6
6	63	9	6
7	**127**	18	18
8	255	30	16
9	511	56	48
10	1023	99	60

Tabella 8.1 Numero di polinomi irriducibili e primitivi per $r \leq 10$.

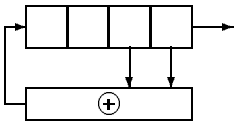
Esempio 8.1. Consideriamo il polinomio irriducibile $p_1(x) = x^4+x^3+x^2+x+1$, che costituisce la rete di retroazione del registro Partendo da stati diversi si possono ottenere cicli diversi, anche se tutti i cicli hanno lo stesso periodo in quanto $p_1(x)$ è irriducibile. Poiché il polinomio divide x^5+1, il periodo ha lunghezza 5 (a parte il "ciclo" contenete lo stato nullo, che è isolato). I $2^4 = 16$ stati sono suddivisi in tre cicli di lunghezza 5 più lo stato nullo. Il polinomio $p_2(x) = x^4+x^3+1$ oltre che essere irriducibile è anche primitivo; il periodo



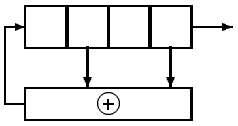
$p_1(x) = x^4 + x^3 + x^2 + x + 1$

0001	1111	0010
1000	0111	1001
1100	1011	0100
0110	1101	1010
0011	1110	0101

Figura 8.7 Cicli del registro associato al polinomio $p_1(x) = x^4 + x^3 + x^2 + x + 1$



$p_2(x) = x^4 + x^3 + 1$



$p_3(x) = x^4 + x^2 + 1$

0001	0101
1000	1010
0100	1101
0010	1110
1001	1111
1100	0111
0110	0011
1011	

0001	1111	1011
1000	0111	1101
0100	0011	0110
1010	1001	
0101	1100	$\frac{(x+1)(x^2+x+1)}{(x^2+x+1)^2}$
0010	1110	
$\frac{1}{(x^2+x+1)^2}$	$\frac{x^2(x+1)}{(x^2+x+1)^2}$	

Figura 8.8 Cicli associati a due diversi *RLSR*

della sequenza è quello massimo, cioè $2^4 - 1 = 15$. Prendiamo ora il polinomio $p_3(x) = x^4 + x^2 + 1$, che è scomponibile in $(x^2 + x + 1)^2$. Essendo riducibile bisogna fare attenzione alla possibilità di semplificazione della (8.11), che riduce il grado del polinomio a denominatore. Se partiamo dagli stati 0001 e 1111 (prime due colonne della figura) non ci sono semplificazioni da fare; $p_3(x)$ divide $x^6 + 1$ e il periodo è 6. Se si parte invece da 1011 si ha una semplificazione che riduce il grado di $f(x)$, che diventa $x^2 + x + 1$; poiché esso divide $x^3 + 1$ il periodo corrispondente è 3. ◯

I registri associati a polinomi primitivi sono in grado di generare sequenze a periodo massimo. Il teorema seguente ci assicura inoltre che tali sequenze sono perfettamente equilibrate per quanto riguarda distribuzione delle tratte e la funzione di autocorrelazione.

Teorema 8.4. *Il periodo di una sequenza generata da un RLSR associato a un polinomio caratteristico primitivo soddisfa i criteri di Golomb* **Gol 1, Gol 2, Gol 3** .

Dim. **Gol 1.** Ciascun stato fra i 2^r-1 possibili (a parte quello nullo) compare una e una sola volta all'interno del ciclo. Interpretando ognuno di essi come intero in base 2 il ciclo tocca tutti gli interi tra 1 e 2^r-1; di questi $2^{r-1}-1$ sono pari e 2^{r-1} sono dispari. Ma ogni stato pari ha 0 come ultimo bit e ogni stato dispari ha 1; poiché la sequenza si ricava dall'ultimo bit ci sono $2^{r-1}-1$ zeri e 2^{r-1} uni.

Gol 2. Per quanto riguarda le tratte si rifletta sul fatto che l'evoluzione degli stati nel registro corrisponde a una finestra di ampiezza r che si sposta (p.es. verso sinistra) all'interno della sequenza. Cominciamo con le tratte di lunghezza maggiore. Se esistesse una tratta unitaria di lunghezza $\geq r+1$ comparirebbero due stati unitari $11\ldots11 = 1^r$ consecutivi, il che è escluso dal fatto che ogni stato compare una sola volta. Dunque le tratte unitarie devono avere lunghezza $\leq r$. C'è inoltre una sola tratta unitaria di lunghezza r perché in caso contrario lo stato unitario comparirebbe due volte all'interno dello stesso periodo. Non ci possono essere tratte unitarie di lunghezza $r-1$, poiché gli stati 01^{r-1} e $1^{r-1}0$ comparirebbero in due occasioni: la prima per la tratta 01^r0 e la seconda per la tratta $01^{r-1}0$. Le tratte unitarie di lunghezza $k \leq r-2$ si possono invece contare riempiendo in un modo arbitrario le $r-k-2$ posizioni α nella finestra di lunghezza r data da $01^k0\alpha^{r-k-2}$. Per le tratte nulle vale lo stesso discorso, con la differenza che non esistono tratte nulle di lunghezza $\geq r$, poiché il registro cadrebbe nello stato nullo e vi permarrebbe indefinitamente; l'assenza di queste tratte fa venir meno il ragionamento precedente in merito alla tratta di lunghezza $r-1$, che è dunque ammissibile. Tanto le tratte unitarie che le nulle sono dunque in numero di

$$1+\sum_{k=1}^{r-2} 2^{r-k-2}$$

Il numero totale di tratte ammonta a

$$2\left(1+\sum_{k=1}^{r-2} 2^{r-k-2}\right) = 2^{r-1}$$

e tra queste quelle di lunghezza k (unitarie e nulle) corrispondono a una frazione pari a

$$\frac{2\cdot 2^{r-k-2}}{2^{r-1}} = 2^{-k}$$

Gol 3. Poniamo $p = 2^r - 1$ e consideriamo le sequenze (infinite a destra) ottenute a partire da diverse posizioni della sequenza del registro

$$\begin{aligned} \boldsymbol{s_1} &= a_1 a_2 \ldots a_p a_1 a_2 \ldots \\ \boldsymbol{s_2} &= a_2 a_3 \ldots a_1 a_2 a_3 \ldots \\ &\vdots \\ \boldsymbol{s_p} &= a_p a_1 \ldots a_{p-1} a_p a_1 \ldots \\ \boldsymbol{s_0} &= 00 \ldots 000 \ldots \end{aligned} \tag{8.16}$$

Si verifica facilmente che esse costituiscono un gruppo abeliano nei confronti dell'operazione somma bit per bit modulo 2, con $\boldsymbol{s_0}$ elemento neutro. L'insieme è inoltre stabile rispetto l'operazione, poiché se $\boldsymbol{s_i}$ e $\boldsymbol{s_j}$ soddisfano entrambe l'equazione di ricorrenza lineare (8.9), essa è soddisfatta anche da $\boldsymbol{s_i} + \boldsymbol{s_j}$, in quanto tale sequenza è determinata dai suoi primi r simboli, che qualunque siano sono i primi r simboli di qualcuna delle $\boldsymbol{s_h}$ ($0 \leq h \leq p$). Il calcolo della funzione di autocorrelazione comporta il calcolo delle coincidenze e delle discordanze tra una certa $\boldsymbol{s_i}$ e una sua traslata $\boldsymbol{s_{i-t}}$, cioè il calcolo del numero di zeri e uni della sequenza somma, che è una sequenza del gruppo. Tenendo conto di **Gol 1** si ricava che

$$C(t) = \frac{(2^{r-1} - 1) - 2^{r-1}}{2^r - 1} = \frac{-1}{2^r - 1} \qquad \forall\, t > 0$$

□

I registri lineari a scorrimento retroazionato associati a polinomi primitivi sono dunque degli eccellenti generatori dal punto di vista della pseudocasualità, poiché soddisfano addirittura i criteri di Golomb. Come tali hanno trovato vasto impiego in molte applicazioni che richiedono sequenze pseudocasuali (si veda ancora il testo di Golomb [40]). Tuttavia essi non possono essere impiegati direttamente in ambito crittologico, poiché manifestano una debolezza strutturale dal punto di vista degli attacchi crittanalitici con testo in chiaro. Tale debolezza è ampiamente prevedibile nella misura in cui il meccanismo di generazione della sequenza è governato da un'equazione di ricorrenza lineare. Per effettuare la completa forzatura è infatti sufficiente avere a disposizione $2r$ bit consecutivi a fronte dei ben $2^r - 1$ del periodo.

Supponiamo di aver attuato un attacco crittanalitico con testo in chiaro e di aver recuperato la seguente porzione di lunghezza $2r$ della sequenza pseudocasuale: $a_{n+1} a_{n+2} \ldots a_{n+2r-1} a_{n+2r}$. L'applicazione della (8.9) ai bit intercettati porta al seguente sistema di r equazioni nelle r incognite $c_1, c_2, \ldots, c_r$

$$\begin{aligned} a_{n+2r} &= c_1 a_{n+2r-1} + c_2 a_{n+2r-2} + \ldots + c_r a_{n+r} \\ a_{n+2r-1} &= c_1 a_{n+2r-2} + c_2 a_{n+2r-3} + \ldots + c_r a_{n+r-1} \\ &\vdots \\ a_{n+r+1} &= c_1 a_{n+r} + c_2 a_{n+r-1} + \ldots + c_r a_{n+1} \end{aligned} \tag{8.17}$$

risolto il quale si ha accesso alla struttura completa del registro. Poiché $2r/(2^r - 1)$ è infinitesimo con $r \to \infty$ è sufficiente acquisire una porzione infinitesima del periodo per ricostruire la rete di retroazione, e dunque l'intera sequenza.

8.2.3 Complessità lineare delle sequenze

La forzatura del cifrario descritta dal sistema (8.17) sarebbe molto più difficile se la lunghezza del registro necessario per ricostruire la sequenza fosse molto elevata, p.es. proporzionale alla lunghezza del periodo.

Def. 8.1. *Si definisce complessità lineare di una sequenza (o equivalente lineare della stessa) la lunghezza del più piccolo registro lineare a scorrimento retroazionato in grado di generare la sequenza.*

Sulla base della definizione, la complessità lineare di una *pn*-sequenza generata da un registro di lunghezza r è ovviamente r. D'altra parte si può attribuire un valore di complessità lineare a *qualunque* sequenza di lunghezza p, anche se essa non è stata generata in modo lineare, poiché nella peggiore delle ipotesi si può usare un *RLSR* di lunghezza p pari alla lunghezza della sequenza e associato al polinomio $x^p + 1$ (che è in pratica un registro chiuso ad anello). Dunque tutte le sequenze sono generabili in modo lineare, e la valutazione della complessità lineare consente di misurare la difficoltà di ricostruire una qualunque sequenza basandosi sull'equazione di ricorrenza lineare (8.9).

Oss. 8.3. La complessità lineare è legata in modo forte alla complessità di Kolmogorov delle sequenze (si veda [11]). In entrambi i casi si tenta una ricostruzione *algoritmica* di una sequenza preassegnata, usando una *macchina di Turing* nel caso della complessità di Kolmogorov, e un *RLSR* in quello della complessità lineare.

La sicurezza nei confronti di un attacco con testo in chiaro si ottiene allora interpretando il requisito **G 3** della sezione 8.2.1 nel senso di richiedere un *elevato valore della complessità lineare* della sequenza generata.

In generale assegnata una sequenza periodica qualunque, anche se con un prefisso aperiodico , questa può essere in ogni caso espressa nei termini della funzione generatrice (8.10) come

$$G(x) = a(x) + \frac{r(x)}{1 - x^p} = a(x) + \frac{q(x)}{p(x)} \tag{8.18}$$

dove il grado di $p(x)$ è l'equivalente lineare e $a(x)$ corrisponde alla parte aperiodica.

Si può fra l'altro osservare che una complessità lineare (mediamente) elevata è una caratteristica peculiare delle sequenze aleatorie. Si consideri infatti l'insieme delle 2^p sequenze di lunghezza p. Si può dimostrare (si veda [71]) che la distribuzione $N_p(L)$ delle complessità lineari L è data dalla seguente relazione

$$N_p(L) = \begin{cases} 2^{\min(2p-2L,2L-1)} & p \geq L > 0 \\ \\ 1 & p > L = 0 \end{cases} \tag{8.19}$$

che consente di ricavare il valor medio e la varianza della distribuzione

$$\begin{aligned} E[L(p)] &= \frac{p}{2} + \frac{4 + r_2(p)}{18} + \epsilon_1(p) \\ Var[L(p)] &= \frac{86}{81} + \epsilon_2(p) \end{aligned} \tag{8.20}$$

dove $r_2(p) = p \mod 2$, mentre $\epsilon_1(p)$ ed $\epsilon_2(p)$ sono infinitesimi. I valori asintotici sono rispettivamente

$$E[L(\infty)] \approx \frac{p}{2} \qquad Var[L(\infty)] \approx \frac{86}{81} \tag{8.21}$$

e vengono approssimati molto bene già per piccoli valori di p.

In sostanza la distribuzione delle complessità è centrata sul valore $p/2$, con una varianza strettissima e praticamente indipendente da p. La tabella 8.2 fornisce i valori della distribuzione per $p \leq 10$

$L(p)$ | $p:$	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	2	2
2		1	4	8	8	8	8	8	8	8
3			1	4	16	32	32	32	32	32
4				1	4	16	64	128	128	128
5					1	4	16	64	256	512
6						1	4	16	64	256
7							1	4	16	64
8								1	4	16
9									1	4
10										1

Tabella 8.2 Distribuzione delle complessità lineari per $p \leq 10$.

Oss. 8.4. Poiché il valore medio (asintotico) della complessità lineare è pari a $p/2$ con una varianza molto bassa, la stragrande maggioranza delle sequenze

hanno complessità molto prossima a $p/2$; per ricostruire una di tali sequenze su base algoritmica si devono specificare $\approx p/2$ bit per la struttura del registro (rete di retroazione) e altrettanti bit per lo stato iniziale. Di conseguenza la ricostruzione algoritmica di una sequenza di lunghezza p richiede (mediamente) circa p bit. Questo significa che le sequenze generate da una sorgente aleatoria, binaria, uniforme, stazionaria e senza memoria non sono mediamente comprimibili, cosa ben nota dall'applicazione a detta sorgente del teorema di Shannon (3.64), che comporta una limitazione inferiore per il tasso pari al valore che l'entropia assume in condizioni di d.p. uniforme, cioè $\log 2^p = p$.

A seguito di queste considerazioni, una complessità lineare elevata diventa un carattere distintivo delle sequenze aleatorie, e come tale bisogna tentare di riprodurlo nella costruzione del determinismo che genera la sequenza pseudocasuale. All'aumentare del valore di p non è tuttavia sufficiente che il solo valore statico della complessità lineare sia elevato; è piuttosto necessario che la sequenza sia dotata di un buon *profilo di complessità* , e cioè che $L(p)$ segua in modo irregolare la retta $L = p/2$. Come esempio si può considerare la complessità lineare relativa alla sequenza di Legendre (a_n), con

$$a_n = \begin{cases} 1 & \text{se } n = 2^j - 1 \\ 0 & \text{altrimenti} \end{cases}$$

la quale ha una complessità lineare elevata, ma un profilo estremamente regolare che tradisce la sua natura funzionale. Si tratta infatti di una "scalinata" con scalini tutti uguali di lunghezza 2 e ad altezze costante. Nella figura 8.9 si vede

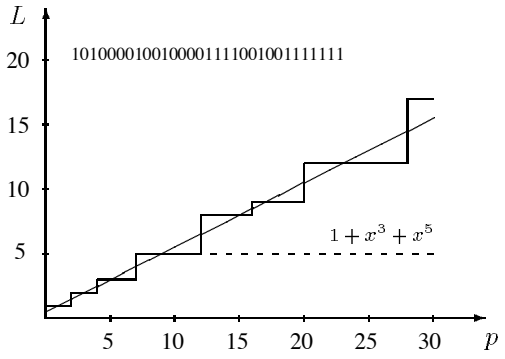


Figura 8.9 Tipico profilo di complessità lineare per una sequenza aleatoria.

il confronto fra un tipico profilo di complessità di una sequenza aleatoria (linea

continua) e quello relativo a una *pn*-sequenza generata dal registro $1 + x^3 + x^5$ di lunghezza 5, per la quale la differenza è ben evidente (linea tratteggiata; nel diagramma è riportata anche la retta $L = p/2$).

Oss. 8.5. Si osservi che una sequenza non è *di per sé* aleatoria o non aleatoria. Qualunque sequenza (anche quella di Legendre) *può* uscire da una sorgente squisitamente aleatoria, solo che essa non ne riassume il comportamento tipico determinato dalla legge debole dei grandi numeri. In un certo senso è proprio la capacità di discostarsi dal proprio comportamento tipico che distingue una sorgente squisitamente aleatoria da una deterministica.

La valutazione della complessità lineare di una sequenza preassegnata, in modo da verificarne l'adeguatezza del profilo di complessità, può essere fatta con un algoritmo molto efficiente introdotto originariamente da Berlekamp, nel contesto della decodifica dei codici BCH, per individuare la struttura del polinomio locatore degli errori. Diamo solo una descrizione dell'algoritmo (in pseudo-Pascal), rimandando all'articolo originale [60] per eventuali approfondimenti.

Algoritmo di Massey-Berlekamp. Sia $a_0 a_1 \ldots a_{N-1}$ la sequenza finita di lunghezza N che si deve analizzare, n la posizione del bit corrente e L il valore corrente della complessità lineare. Poniamo inoltre $f(x) = 1 + c_1 x + c_2 x^2 + \ldots + c_L x^L$

1. $f(x) := 1 \quad q(x) := 1 \quad m := 1 \quad L := 0 \quad n := 0$
2. **if** $n = N$ stop **else** compute
$$\delta = a_n + \sum_{i=1}^{L} c_i a_{n-i}$$
3. **if** $\delta = 0$ **then** $m := m + 1$ and go to 6.
4. **if** $\delta \neq 0$ and $2L > n$ **then**
$$f(x) := f(x) + x^m q(x) \qquad m := m + 1$$ and go to 6.
5. **if** $\delta \neq 0$ and $2L \leq n$ **then**
$$t(x) := f(x)$$
$$f(x) := f(x) + x^m q(x)$$
$$L := n + 1 - L \qquad q(x) := t(x) \qquad m := 1$$
6. $n := n + 1$ and return to 2.

La tabella 8.3 mostra il risultato dell'applicazione dell'algoritmo alla sequenza 101001; nella colonna *RLSR* si vede la struttura corrente del registro mentre nella colonna L si può valutare il profilo della complessità lineare.

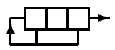
n	L	$f(x)$	$RLSR$	m	$q(x)$	a_n	δ
0	0	1		1	1	1	1
1	1	$1+x$		1	1	0	1
2	1	1		2	1	1	1
3	2	$1+x^2$		1	1	0	0
4	2	$1+x^2$		2	1	0	1
5	3	1		1	$1+x^2$	1	1
6	3	$1+x+x^3$		2	$1+x^2$		

Tabella 8.3 Esempio d'applicazione dell'algoritmo di Massey-Berlekamp.

8.2.4 Alcuni esempi di cifrari a flusso

La debolezza strutturale dei *RLSR* nei confronti degli attacchi con testo in chiaro ne impedisce l'impiego diretto come generatori di sequenze pseudocasuali. Tuttavia la loro semplicità di attuazione, il loro costo estremamente contenuto e le ottime caratteristiche statistiche delle sequenze generate costituiscono dei punti di forza che suggeriscono di usare questi dispositivi come elementi di base nella costruzione dei generatori. Il problema è essenzialmente quello di modificare le sequenze in modo da distruggerne la linearità, facendo così aumentare in modo significativo la complessità lineare delle stesse. Si noti però che tale "distruzione" deve essere in qualche misura controllata, nel senso che per la nuova sequenza non lineare si devono comunque poter garantire le caratteristiche statistiche analizzate precedentemente (frequenza delle tratte, autocorrelazione ecc.).

Una prima possibilità è quella di costruire dei cicli costituiti dalla *giustapposizione* di tutte e sole le 2^r r-ple possibili (disposte in modo aleatorio all'interno del ciclo), con $r \to \infty$. Così facendo il ciclo di lunghezza $r \cdot 2^r$ ha delle buone prestazioni per quanto riguarda la sua pseudocasualità, in quanto soddisfa i criteri estesi di Golomb analizzati nel paragrafo8.2.1. Inoltre la complessità lineare è proporzionale alla lunghezza del periodo [29, 30]. Così facendo rimane però aperto il problema di meccanizzare la concatenazione delle r-ple, se si vuole passare a un generatore effettivo. Si può ricorrere allora all'introduzione diretta di non linearità nelle sequenze generate da un *RLSR*, secondo uno dei seguenti quattro metodi:

1. usare direttamente una rete di retroazione non lineare, costruendo così un registro *non lineare*;

2. comporre in modo non lineare le sequenze in uscita da diversi *RLSR*;
3. filtrare con una funzione non lineare l'uscita di un singolo *RLSR*;
4. usare le sequenze in uscita da un *RLSR* per comandare l'orologio interno di sincronismo di altri *RLSR*.

Analizziamo brevemente queste tecniche fornendo qualche esempio, e rimandando a letture più specialistiche per gli approfondimenti del caso.

Registri non lineari

La struttura generale di un registro non lineare di lunghezza r è riportata in figura 8.10; in essa la rete di retroazione è costituita da una fra le possibili 2^{2^r}

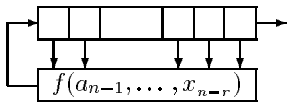


Figura 8.10 Struttura di un registro non lineare.

funzioni Booleane.

L'equazione di ricorrenza (non lineare) è in questo caso data dall'espressione

$$a_n = f(a_{n-1}, a_{n-2}, \ldots, a_{n-r}) \qquad n \geq r \tag{8.22}$$

Lo studio di questo tipo di registri non ha ancora consentito di stabilire criteri generali per costruire sequenze pseudocasuali d'interesse crittografico. L'unica eccezione è costituita dalle sequenze di de Bruijn, che sono sequenze a ciclo massimo di lunghezza 2^r (la non linearità consente di sfuggire dallo stato nullo, che non è dunque necessariamente isolato) e per le quali esiste una letteratura molto vasta (si veda per esempio [35]). Un esempio semplice lo si ha usando la funzione $f(x_1, x_2, x_3) = 1 \oplus x_2 \oplus x_3 \oplus x_1 x_2$ con $r = 3$, che genera la sequenza 00011101. Si può dimostrare che in tal caso il registro passa per tutti gli stati una sola volta, e che in generale le sequenze di de Bruijn hanno ottime caratteristiche statistiche. Fra l'altro esse si possono ottenere direttamente dalle *pn*-sequenze semplicemente aggiungendo uno 0 alla tratta nulla di lunghezza $r - 1$. Ciò le rende in pratica estremamente deboli dal punto di vista crittografico.

Registri composti

La struttura generale di un generatore basato sulla composizione non lineare di più registri è quella riportata in figura 8.11(a). L'esempio più immediato di

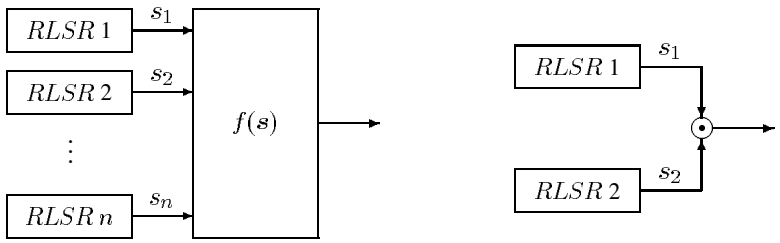


Figura 8.11 (a) Struttura di un registro composto; (b) registro prodotto.

composizione non lineare tra le sequenze generate da due *RLSR* primitivi, di lunghezza rispettivamente r ed s, è quello di effettuare il prodotto bit per bit delle due sequenze (fig. 8.11(b)). In questo modo la complessità lineare diventa pari a $r \cdot s$ e il periodo pari al prodotto dei due periodi, perlomeno nelle ipotesi che $(r, s) = 1$ (sempre sottintesa in tutte le realizzazioni con più registri). Naturalmente la regolarità statistica viene persa, poiché il prodotto tra due bit fornisce 0 in tre casi su quattro. Una soluzione al problema è fornito dal *generatore di Geffe*, costituito da tre *RLSR* primitivi e con lunghezze prime tra loro. La funzione usata è

$$f(x_1, x_2, x_3) = x_1 x_2 \oplus x_3(1 \oplus x_2)$$

e si potrebbe dimostrare che il generatore ha un buon bilanciamento delle tratte e una complessità lineare apprezzabile ($> \max(r_2 r_1, r_2 r_3)$)[39]. Tuttavia il sistema è debole nei confronti degli *attacchi di correlazione*, che in presenza di correlazione fra la sequenza di uscita e la sequenza di uno dei registri consente di ricavare le condizioni iniziali degli stessi (vedi [79] o [71] per un'analisi approfondita). In generale per tutti i registri di questa classe bisogna porre particolare attenzione agli attacchi a correlazione, poiché sono molto spesso un efficace metodo di forzatura.

Generatore con filtro non lineare

Il principio di funzionamento è illustrato in figura 8.13(a); si vede che lo stato del registro costituisce l'ingresso per una funzione Booleana opportuna la cui uscita determina la sequenza. La struttura è simile a quella del registro a retroazione non lineare, con la differenza che in questo caso l'evoluzione degli stati dipende dalla struttura dalla rete di lineare. Un esempio interessante è dato dal *multiplatore* di figura 8.13(b). Ci sono due registri *RLSR* 1 e 2, di lunghezza

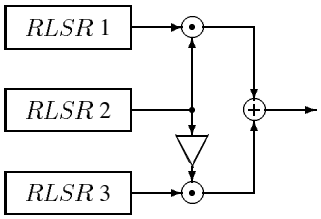


Figura 8.12 Generatore di Geffe.

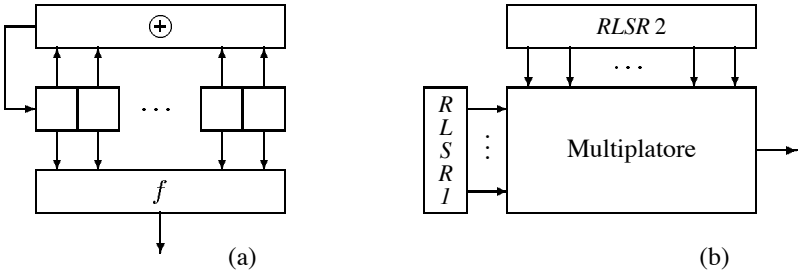


Figura 8.13 (a) Generatore con filtro non lineare (b) Il multiplatore.

rispettivamente h e m, e lo stato del primo è la rappresentazione binaria dell'indirizzo corrispondente alla cella del secondo registro che deve mandare in uscita il bit. Il dispositivo multiplatore si occupa di effettuare il corretto indirizzamento su *RLSR* 2. Perchè tutto funzioni deve essere $m \geq 2^h$, e sotto ipotesi minime (m e h primi tra loro), posto $t = 2^h - 1$ e $q = 2^m - 1$, si ottengono le seguenti prestazioni

periodo	$p = (2^h - 1)(2^m - 1)$
compl. lin.	$L = m(2^h - 1)$
funz. autocorr. media	$E[C(t)] = \dfrac{t-q}{q(qt-1)} \approx \dfrac{1}{q^2} - \dfrac{1}{qt}$
funz. autocor.	$C(t) \approx \dfrac{-1}{2^m - 1}$ se $m >> h$

Registri comandati in sincronismo

Per concludere diamo un brevissimo cenno su un principio alternativo alla manipolazione non lineare a posteriori di *pn*-sequenze generate con *RLSR*. L'idea è quella di agire sugli orologi di sincronizzazione che regolano il funzionamento dei registri, o in modo da provocare delle interruzioni (con ripetizione

dell'ultima uscita disponibile), oppure selezionando solamente una sottosequenza della sequenza data. In ogni caso l'effetto è quello di distruggere la linearità strutturale delle sequenze originarie. Un esempio in tal senso è riportato in figura 8.14. Gli impulsi di sincronizzazione fanno funzionare il registro di co-

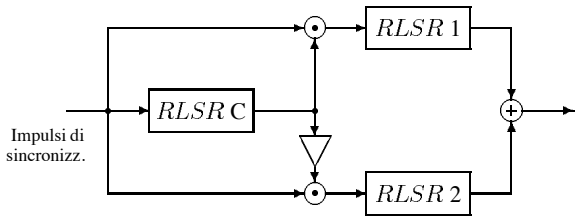


Figura 8.14 Generatore comandato in sincronismo.

mando *RLSR* C. Se la sua uscita è 1 viene attivato *RLSR* 1, che manda in uscita il bit generato; se esce 0 viene attivato *RLSR* 2. Quando un registro rimane inattivo, cioè non riceve impulsi di sincronizzazione, si suppone che continui a emettere sempre l'ultimo bit generato. La sequenza di uscita è data dalla somma bit per bit mod. 2 delle due sequenze.

Un'altra possibilità per comandare in sincronismo una sequenza è quello di estrarre dalla stessa *solo* i bit che corrispondono agli 1 di una seconda sequenza. Per esempio siano 011100101110010 la sequenza di sincronizzazione e 000100110101111 la sequenza comandata. Si ottiene

$$\begin{array}{cccccccccccccccc} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{ccccccccccccc} 0 & 0 & 1 & & & 1 & & 0 & 1 & 0 & & & 1 \end{array}$$

cioè 00110101. Entrambi questi sistemi manifestano buone doti di pseudocasualità e di resistenza agli attacchi crittanalitici noti.

Per completezza citiamo le referenze bibliografiche di qualche altro metodo che ha suscitato l'interesse degli specialisti. Oltre al *Knapsack generator* [71], basato sul ben noto problema di ottimizzazione che ha ispirato anche il cifrario a chiave pubblica di cui parleremo nel paragrafo 9.3.1, vale la pena di citare il sistema *SEAL* [70], coperto da brevetto, e l'impiego del *DES* come generatore di sequenze pseudocasuali. In tal caso si possono usare gli stati di un *RLSR* di 64 bit come ingresso al *DES*, che lavora con una chiave di 56 bit. A ogni ciclo di funzionamento del registro il *DES* fornisce in uscita un blocco di 64 bit, e la concatenazione dei blocchi costituisce la sequenza pseudocasuale da sommare al flusso dei messaggi. Lo schema di base si può complicare secondo alcune varianti, una delle quali prevede la retroazione dell'uscita *DES* con l'ingresso (ma per i dettagli si veda [5]).

Capitolo 9
Cifratura a chiave pubblica

La crittologia a chiave pubblica, le cui linee essenziali sono state anticipate nel capitolo introduttivo, sfrutta l'esistenza di funzioni speciali denominate *unidirezionali*, delle quali si può dare la seguente definizione euristica

Def. 9.1. *Una funzione $f : X \to Y$ si dice unidirezionale se $\forall x \in X$ è computazionalmente "facile" calcolare $f(x)$, ma per quasi tutti gli $y \in f(X)$ il problema di ricavare un qualunque x tale che $f(x) = y$ è "intrattabile".*

Nella definizione si usano i termini "facile" e "intrattabile" nel senso loro attribuito nel contesto della *teoria della complessità*, di cui daremo breve cenno nella sezione 9.1. Per il momento accontentiamoci di considerare "facili" tutte le computazioni effettivamente attuabili in un tempo ragionevole, e "intrattabili" i problemi per i quali trovare la soluzione richiederebbe un tempo di calcolo esponenziale, in grado cioè di debordare asintoticamente le capacità di calcolo di qualunque elaboratore. La funzione è in questo caso associata a un elevato tasso di asimmetria nella sua computazione, a seconda che la si percorra in senso diretto come $f(x)$ o in senso inverso come $f^{-1}(y)$.

Nella definizione c'è inoltre il termine "quasi", riferito alla numerosità degli y, che porta un ulteriore elemento d'imprecisione. Ciò dipende dal fatto che anche problemi considerati difficili, o intrattabili per usare il nuovo termine, possono comunque avere delle istanze per le quali il calcolo risulta molto semplice, e la funzione unidirezionale diventa facilmente invertibile. Quando si affida la sicurezza di un sistema di cifratura all'intrattabilità di un certo problema bisogna dunque far attenzione a non ricadere nelle istanze "facili" dello stesso.

In certi casi l'inversione della funzione unidirezionale diventa particolarmente semplice *per tutte le istanze del problema* non appena si disponga di un'informazione suppletiva.

Def. 9.2. *Una funzione unidirezionale f da X a Y si dice ad accesso se, a seguito di un'informazione suppletiva, $\forall y \in Y$ diventa computazionalmente "facile" ricavare x tale che $f(x) = y$.*

(nella letteratura internazionale si usano rispettivamente i termini *one-way function* e *trapdoor one-way function*) .

Tutta la crittografia a chiave pubblica, o asimmetrica , basa il proprio funzionamento sull'esistenza di certe funzioni unidirezionali, le più importanti delle quali verranno passate in rassegna nella sezione 9.2.

In uno schema di cifratura a chiave pubblica abbiamo in generale un insieme di utenti $X, Y, Z, \ldots$. Ciascuno di essi genera per conto proprio, o acquisisce da un *Autorità Garante* (o semplicemente *Garante*), una coppia di funzioni di cifratura e di decifrazione che appartengono alla classe delle funzioni unidirezionali (ad accesso). Usiamo la terminologia C_x e D_x per denotare rispettivamente la chiave di cifratura e decifrazione dell'utente X. Si deve inoltre supporre che la conoscenza della chiave di cifratura C_x non consenta di ricavare la chiave di decifrazione $D_x = C_x^{-1}$, poiché l'unidirezionalità della funzione C_x implica la necessità per il crittanalista di risolvere un problema intrattabile.

Così facendo le chiavi di cifratura $C_x, C_y, C_z \ldots$ possono essere divulgate, per esempio mediante la pubblicazione di un elenco ufficiale delle chiavi asseverato dal Garante, rendendole disponibili a tutti gli utenti della rete. Viceversa le chiavi di decifrazione $D_x, D_y, D_z \ldots$ devono rimanere segrete e custodite privatamente dagli utenti. Poiché la cifratura avviene *solo* attraverso le chiavi di cifratura, non c'è necessità di far circolare le chiavi private di decifrazione, risolvendo in tal modo il problema della distribuzione delle chiavi.

Se dunque l'utente X vuole spedire un messaggio riservato m all'utente Y, secondo lo schema della (7.3), egli deve consultare l'elenco pubblico asseverato delle chiavi di cifratura, leggere la C_y e applicare la trasformazione $C_y(m)$, ottenendo il crittogramma c. In ricezione l'utente legittimo Y, che conosce (ed è il solo) la chiave inversa D_y, eseguirà l'operazione $D_y(c) = D_y C_y(m)$ sul crittogramma ricevuto, riottenendo il messaggio. Tutto il procedimento si basa sul fatto che il problema di ricavare la chiave inversa da quella diretta, che chiamiamo per comodità *problema inverso*, è computazionalmente *intrattabile*.

Oss. 9.1. Un sistema di crittografia a chiave pubblica non è in ogni caso in grado di fornire una *sicurezza incondizionata*. Infatti avendo intercettato un crittogramma $c = C_x(m)$ trasmesso da X, un crittanalista può in linea di principio usare la chiave pubblica C_x per cifrare uno alla volta tutti i possibili messaggi che X può generare, fin quando trova l'unico m tale che $C_x(m) = c$.

La presenza di un Garante è essenziale per il buon funzionamento del sistema, anche se la sua natura e la sua azione di controllo possono assumere connotazioni molto diverse.

Oss. 9.2. La sicurezza di un qualunque sistema crittografico, a chiave segreta o pubblica che sia, necessità di una inizializzazione basata su un'asseverazione primitiva di una qualche informazione che deve essere poi condivisa da almeno due soggetti. Solo sull'autenticità di quest'informazione primitiva si può innescare qualunque processo di trasmissione sicura o di autenticazione dei soggetti coinvolti nella stessa.

9.1 Complessità computazionale

La sicurezza di un sistema a chiave pubblica è legata al tasso di unidirezionalità della funzione associata al sistema, cioè alla complessità computazionale del problema di calcolare la chiave inversa D_k a partire da quella diretta C_k. Per tale complessità bisogna trovare una misura idonea, e la *teoria della complessità* offre alcuni strumenti in questa direzione, consentendo di effettuare una classificazione degli algoritmi che portano alla soluzione di vari problemi in base alle risorse (temporali o di memoria) che essi richiedono, formulando così una valutazione della loro efficienza.

In questa sezione tratteremo il tema della complessità computazionale in un modo molto informale, ma sufficiente per comprendere i punti di forza (e di debolezza) dei sistemi basati sulle chiavi asimmetriche. Per un approccio rigoroso ed esauriente al problema rimandiamo il lettore al testo di Garey-Johnson [37], universalmente riconosciuto come pietra miliare della teoria.

Definiamo informalmente *algoritmo* una procedura a passi per ottenere la soluzione di un certo *problema* computazionale. Alcuni semplici esempi potrebbero essere il prodotto tra due interi in rappresentazione binaria di lunghezza n bit, la soluzione di un sistema di m equazioni in n incognite binarie, l'ordinamento per ordine decrescente di un insieme di n interi ecc. Fissato che sia il problema, una sua *istanza* è la specificazione effettiva dei parametri del problema (p.es che si vogliono moltiplicare tra loro gli interi 3 e 7, o che l'insieme degli interi da ordinare è composta dai numeri $5, 9, 2, 34$).

Supponiamo di aver dato una descrizione esauriente del problema, avendo identificato un modo idoneo per codificarlo come ingresso dell'algoritmo. La complessità computazionale dell'algoritmo è allora legata intuitivamente al numero di passi necessari per approdare alla soluzione, fissata che sia la lunghezza n dell'ingresso. Se $f(n)$ è la funzione che esprime il numero di passi elementari in funzione di n, la modalità di crescita di $f(n)$ è fondamentale per comprendere se il problema possa o meno essere risolto in modo efficiente (in pratica in un tempo ragionevole). Più precisamente si usa introdurre una funzione nella forma $O(g(n))$, denominata *ordine di grandezza*, con il seguente significato

$$f(n) = O(g(n)) \quad \text{se} \quad \exists\, \alpha, \bar{n} : f(n) \leq \alpha \left|g(n)\right| \quad \text{per} \quad n \geq \bar{n} \tag{9.1}$$

La notazione $O(g(n))$ esprime il fatto che la funzione $f(n)$ cresce asintoticamente con una velocità non maggiore di $g(n)$; essa sottintende inoltre che non si è interessati alla esatta quantificazione dei passi, quanto piuttosto all'ordine d'infinito presentato dalla $f(n)$. D'altra parte la quantificazione esatta dipende in modo diretto dal tipo di codifica usata per esprimere l'istanza del problema e

dal modello di computazione impiegato. Asintoticamente le differenze riguardanti la diversa codifica diventano trascurabili, e l'intero peso computazionale si aggrega attorno all'ordine di grandezza $O(g(n))$.

Gli algoritmi si possono dunque classificare sulla base del loro ordine di grandezza, tenendo conto del *massimo ordine necessario* per risolvere una qualunque istanza del problema. Accade infatti che per un prefissato algoritmo ci siano istanze che si risolvono più facilmente di altre, cioè con un ordine minore. Un algoritmo si dice allora *polinomiale* (o in tempo polinomiale) se $f(n) = O(n^r)$ per qualche valore di r. Se $r = 0$ si parla di complessità *costante*, che si denota anche come $O(1)$ (la complessità non dipende dalla lunghezza dell'ingresso); viceversa per $r = 1, 2, 3, \ldots$ si ha rispettivamente complessità *lineare, quadratica cubica* ecc.

Qualunque algoritmo per il quale la complessità dell'istanza peggiore non possa essere limitata superiormente in modo polinomiale è detto a *complessità esponenziale*.

Da un punto di vista asintotico la differenza tra complessità polinomiale ed esponenziale è drammatica, tale cioè da rendere praticamente inutile qualunque sforzo tecnologico per velocizzare il numero di operazioni elementari per secondo effettuate dagli elaboratori reali. La misura di tale affermazione ci viene dalla consultazione della tabella 9.1, nella quale si ipotizza l'uso di un elaboratore in grado di effettuare 10^9 operazioni elementari al secondo per trattare con algoritmi di diversa complessità un ingresso di lunghezza pari a $n = 10^6$. La

$f(n)$	n	n^2	n^3	2^n
tempo (a 10^9 op/s)	10^{-3} s	16.6 min.	31.7 anni	$3.14\ 10^{301004}$ mld di anni

Tabella 9.1 Tempi di calcolo per algoritmi di diverse classi di complessità .

teoria della complessità usa come modello computazionale la cosiddetta *Macchina di Turing* [37, 2], che è una macchina a stati finiti dotata di un nastro di memoria infinito, sul quale si possono leggere e scrivere simboli appartenenti a un alfabeto finito. Essa è un modello computazionale astratto di un qualunque calcolatore reale, nel senso che si può dimostrare che un qualunque calcolatore reale può risolvere *solo* i problemi che possono essere risolti da una macchina di Turing.

Se per un problema esiste un algoritmo in tempo polinomiale in grado di risolvere tutte le possibili istanze diremo che esso è *trattabile*; viceversa chiameremo *intrattabili* i problemi per i quali non sono noti algoritmi in tempo polinomiale. Tutti i problemi risolubili in tempo polinomiale da una macchina di Turing

deterministica vengono aggregati nella cosiddetta *classe di complessità* di tipo *P*.

Si osservi che la teoria della complessità fa riferimento essenzialmente ai cosiddetti *problemi di decisione*, per i quali si hanno risposte del tipo SÌ - NO. Anche se questo approccio può sembrare limitativo in pratica non lo è, poiché molti problemi computazionali possono essere trasformati in tempo polinomiale (e quindi in modo efficiente) in istanze di altrettanti problemi di decisione. Ciò significa che trovare un algoritmo efficiente per il problema di decisione associato comporta l'individuazione implicita di un algoritmo efficiente per il problema computazionale primitivo (e viceversa).

Introduciamo ora altre due classi di importanza fondamentale, dandone una definizione informale.

Def. 9.3. *La classe computazionale NP è costituita dall'insieme di tutti i problemi di decisione per i quali è possibile verificare in tempo polinomiale la correttezza di una risposta di tipo SÌ disponendo di un'informazione suppletiva chiamata certificato.*

La sua complementare è la classe co-NP, costituita dall'insieme di tutti i problemi di decisione per i quali è possibile verificare in tempo polinomiale la correttezza di una risposta di tipo NO (sempre disponendo del certificato).

In altre termini i problemi nella classe *P* sono trattabili, mentre quelli della classe *NP* hanno la proprietà che è trattabile la verifica della correttezza di una certa soluzione.

Come esempio concreto di questi concetti possiamo citare il problema della scomponibilità di numero intero n in fattori distinti del tipo $n = ab$, con $a, b > 1$. In questo caso l'istanza è data dal valore di n, mentre il certificato che consente di accertare la divisibilità è dato dalla conoscenza di uno dei due divisori, p.es. b. Disponendo di n e di b è sufficiente effettuare la divisione per accertare in tempo polinomiale la scomponibilità di n.

Ogni problema della classe *P* appartiene anche alle classi *NP* e *co-NP*, e dunque vale sicuramente l'inclusione $P \subseteq NP$ e $P \subseteq$ *co-NP*. D'altra parte non è noto a tutt'oggi se le due inclusioni siano strette o meno, poiché non è stata ancora dimostrata l'esistenza di un problema che appartenga alla classe *NP* (*co-NP*), ma *non* alla classe *P*. Ciò è dovuto essenzialmente al fatto che la non conoscenza di algoritmi polinomiali per un certo problema non esclude il fatto che essi possano esistere.

Tra i vari problemi *NP* ne sono stati scoperti alcuni che posseggono la seguente interessante proprietà: qualunque problema della classe *NP* può essere ricondotto a uno qualunque di questi ultimi in un tempo polinomiale. Essi vengono aggregati in una classe computazionale speciale, la classe dei problemi *NP-completi* . Un esempio in tal senso è dato dal *problema della somma parziale*,

che si può enunciare nella sua forma più semplice nel modo seguente: assegnato un insieme di interi positivi $A = \{a_1, a_2, \ldots, a_n\}$ e un intero S, determinare se esiste un sottinsieme di A che ha come somma S. Il problema è chiaramente in *NP*, poiché per ogni sottinsieme è possibile verificare in tempo polinomiale se esso sommi o meno a S. D'altra parte non sono noti algoritmi polinomiali per risolverlo, e dunque non è noto se esso sia anche un membro della classe *P*. Il problema della somma parziale sembra quindi essere di complessità esponenziale, e dunque essenzialmente intrattabile.

Sulla base della definizione di classe *NP-completa* è evidente che se si riuscisse a individuare un algoritmo polinomiale per risolvere un qualunque problema *NP-completo*, allora *tutti* i problemi *NP* starebbero anche in *P*, risultando quindi *P=NP*. D'altra parte la dimostrazione che anche uno solo tra i problemi *NP-completi* è intrattabile implicherebbe l'intrattabilità di tutti.

La figura 9.1 riassume in modo visivo le relazioni di inclusione che sembrano più plausibili, anche se occorre ricordare che non esistono al momento prove dell'esistenza di inclusioni strette. La situazione estrema, ritenuta poco verosimile dai maggiori esperti del settore, è che tutto collassi in un unico insieme con *P=NP=co-NP*.

Da un punto di vista degli algoritmi crittografici è chiaro che bisogna guardare

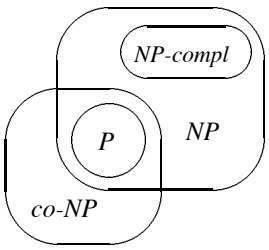


Figura 9.1 Probabili relazioni d'inclusione tra classi di complessità .

a quelle funzioni di cifratura per le quali l'inversione tentata dal crittanalista imponga la necessità di trovare una soluzione a un problema intrattabile, cioè non appartenente alla classe *P*. Viceversa l'inversione operata con l'informazione suppletiva costituita dalla chiave deve risultare attuabile in tempo polinomiale. Se parliamo in termini di complessità, possiamo affermare che il problema della crittanalisi sta in *NP*; infatti disponendo della chiave di cifratura ed essendo la cifratura eseguibile in tempo polinomiale, un crittanalista può verificare in tempo polinomiale se la cifratura di un messaggio stimato porta al crittogramma intercettato. D'altra parte, per lo stesso motivo addotto precedentemente, la decrittazione sta anche in *co-NP*, poiché si può verificare in tempo polinomiale che il messaggio stimato *non* viene cifrato nel crittogramma intercettato.

Dunque, il problema della crittanalisi sta nell'intersezione $NP \cap co-NP$. Ciò ha un'implicazione rilevante, poiché questo fatto *tende a escludere che il problema della crittanalisi sia* $NP-completo$. Quest'ipotesi forzerebbe infatti l'uguaglianza $NP = co-NP$, che al momento non sembra verosimile.

Agganciare un cifrario a un problema *NP-completo* non costituisce tuttavia una garanzia di pieno successo. Ciò accade anche per i seguenti due motivi

1. anche se il problema è *NP-completo* bisogna far attenzione a scegliere i parametri del problema in modo da non incappare in una delle possibili istanze "facili" dello stesso, poiché ciò decreterebbe in pratica la forzatura del sistema (quantomeno di quello che adotta detti parametri);

2. le operazioni di cifratura/decifrazione del sistema devono essere in qualche modo riconducibili al problema intrattabile in oggetto; tuttavia i meccanismi attraverso i quali viene attuato questo legame potrebbero manifestare dei punti di debolezza strutturale, attaccabili dal crittanalista, che di fatto abbassano il livello di complessità globale dell'operazione di decrittazione. Un caso emblematico in tal senso, passato alla storia, è quello del cifrario basato sul problema della somma parziale, (vedi paragrafo 9.3.1), forzato qualche anno dopo la sua presentazione nonostante il problema appartenga di per sé alla classe degli *NP-completi*.

Bisogna dunque distinguere fra complessità del problema associato e complessità dell'operazione di decrittazione, la quale potrebbe essere di gran lunga inferiore.

Oss. 9.3. In generale il punto debole nell'applicazione della teoria della complessità alla crittografia è che essa non offre solitamente criteri di complessità minima garantita, ma piuttosto criteri di complessità massima. Così l'appartenenza alla classe *NP* stabilisce solo che *esistono* istanze del problema per le quali non sono noti algoritmi polinomiali, ma non esclude che vi siano altre istanze (che potrebbero essere anche numericamente molto consistenti) che possono essere risolte in tempo polinomiale.

Oss. 9.4. Se in futuro si dovesse riuscire a provare che *P=NP* tutti i cifrari a chiave pubblica attualmente noti diventerebbero insicuri.

Accanto agli algoritmi di tipo deterministico, inquadrati nelle classi che abbiamo appena visto, sono stati introdotti recentemente *algoritmi di tipo probabilistico*; la loro esecuzione è deterministica in tutti i passi, tranne che in un numero finito di essi dove le decisioni vengono prese su base aleatoria. La differenza fondamentale è che un algoritmo probabilistico *può fornire delle risposte sbagliate* al problema decisionale al quale è dedicato, solo che la ripetizione dell'algoritmo per un numero adeguato di volte consente di ridurre la probabilità d'errore a valori arbitrariamente piccoli. In linea di massima si può dunque

avere in tempo polinomiale una risposta di tipo SÌ (o NO) affetta da un errore che può essere reso arbitrariamente piccolo.

È ovvio che da un punto di vista pratico disporre di un algoritmo probabilistico, affetto da una probabilità di errore di p.es. 2^{-100}, e che fornisce delle risposte in tempo polinomiale a problemi intrattabili, è un'ipotesi estremamente attraente. Si consideri tuttavia che sfortunatamente gli algoritmi probabilistici associati a problemi per i quali non sono noti algoritmi polinomiali sembrano essere piuttosto rari.

Nel seguito (sez. 9.2.4) analizzeremo un rimarchevole esempio di algoritmo probabilistico per la verifica di primalità di un un numero intero assegnato.

9.2 Funzioni unidirezionali d'interesse crittografico

Presentiamo ora un elenco (parziale) di problemi computazionali ritenuti di complessità non polinomiale, che sono stati impiegati concretamente e massicciamente in ambito crittografico. Da un certo punto di vista costituiscono i mattoni base per l'attuazione di obiettivi crittografici di varia natura, che spaziano dalla cifratura a chiave pubblica, all'autenticazione, alla firma numerica e allo scambio di chiavi segrete.

9.2.1 Il problema delle somme parziali

Il problema delle somme parziali (*PSP*), ben noto e studiato nell'ambito della *Ottimizzazione Combinatoria* (si veda p.es. [73]) fu sfruttato per lo schema di crittografia a chiave pubblica proposto da Merkle e Hellman nel 1978. Anche se questo cifrario in particolare si è alla fine dimostrato insicuro (fu forzato, a partire dal 1982, ad opera di Shamir), nel seguito (1988) fu proposta una variante che ha resistito fino al 1998, cioè lo schema di *Chor-Rivest*. E' dunque opportuno descrivere a parte il problema.

PSP Assegnato un insieme di interi positivi $A = \{a_1, a_2, \ldots, a_n\}$ e un intero S, determinare se esiste un sottinsieme di A che ha come somma S, cioè se esiste un n-pla $x_1, x_2, \ldots, x_n$, con $x_i \in \{0, 1\}$, tale che

$$\sum_{i=1}^{n} x_i a_i = S \tag{9.2}$$

Ricordiamo ancora che è stato dimostrato che il *PSP* è un problema *NP- completo*; inoltre esso è computazionalmente equivalente al problema di individuare gli effettivi elementi a_i che entrano nel sottinsieme che somma a S.

Il *PSP* offre anche l'opportunità di porre in rilievo l'importanza dell'osservazione 9.3, in merito all'esistenza di istanze "deboli" del problema; esse consentono infatti l'individuazione di algoritmi polinomiali per la loro risoluzione a dispetto del fatto che il problema sia *NP-completo*. Un esempio in tal senso è dato dal caso in cui si dia agli elementi dell'insieme A un valore *supercrescente*, tale cioè che

$$a_j > \sum_{i=1}^{j-1} a_i \qquad 2 \leq j \leq n \tag{9.3}$$

Se ciò accade il problema delle somme parziali si risolve con un ordine di complessità polinomiale (anzi lineare) usando un algoritmo ovvio, del tutto analogo a quello che s'impiega per le trasformazioni tra basi numeriche. Esso consiste nel confrontare successivamente S con $a_n, a_{n-1}, \ldots$, fin quando si trova $j : S \geq a_j$; a questo punto si ha $x_i = 0$ per $i > j$, $x_j = 1$ e $S \leftarrow S - a_j$. I successivi confronti si ripetono a partire da $i = j - 1$ e fino a completamento dell'ennupla.

Dal punto di vista crittografico, come vedremo nel paragrafo 9.3.1, questa istanza "facile" viene impiegata per consentire una decifrazione in tempo polinomiale per lo schema di *Merkle-Hellman*. Ribadiamo ancora che l'inesistenza di algoritmi efficienti per la soluzione del *PSP* (i migliori algoritmi sono esponenziali) non fornisce garanzie a priori sulla sicurezza dei corrispondenti cifrari da esso derivati.

9.2.2 Il logaritmo discreto

Il logaritmo discreto è stato il primo esempio d'impiego delle funzioni unidirezionali in ambito crittografico, poiché risale al già citato articolo [25] di Diffie e Hellman, che diede avvio alla rivoluzione della crittografia a chiave pubblica. In esso si presenta un metodo che consente a due utenti di scambiarsi in modo sicuro una chiave segreta, da usare abbinata a un sistema crittografico a chiave simmetrica (come p.es. il *DES*). Il logaritmo discreto è usato anche per il sistema crittografico a chiave pubblica di *ElGamal* e per lo schema di firma numerica dello stesso Autore.

In questo caso la funzione unidirezionale è l'elevamento a potenza degli elementi di un gruppo ciclico finito (funzione diretta); la corrispondente funzione inversa, computazionalmente intrattabile, è data dal calcolo del logaritmo di un certo elemento del gruppo. A prima vista potrebbe sembrar strano che una funzione come il logaritmo, facilmente calcolabile sull'asse reale, diventi addirittura una funzione intrattabile quando confinata all'interno di una struttura finita; tuttavia ad oggi non sono noti algoritmi deterministici asintoticamente efficienti. Ciò significa che, quando la dimensione del gruppo tende a diventare molto

grande, anche i migliori algoritmi disponibili fanno esplodere il numero di passi necessari per le istanze più difficili.

Per semplicità , nella nostra trattazione faremo riferimento al campo finito $\boldsymbol{Z_p}$ con p primo, costituito dagli interi delle classi di resto modulo p. In $\boldsymbol{Z_p}$ valgono le ordinarie operazioni di somma e moltiplicazione, previa riduzione $\bmod p$, ed esiste almeno un elemento primitivo α con le potenze del quale si ottengono tutti gli elementi non nulli del campo; questi ultimi costituiscono il gruppo moltiplicativo $\boldsymbol{Z_p^*}$ di ordine $p-1$.

Il *problema del logaritmo discreto* (*PLD*) si formula allora nel modo seguente:

PLD. Assegnati p primo, α elemento primitivo di $\boldsymbol{Z_p}$ e $\beta \in \boldsymbol{Z_p^*}$ si trovi l'unico intero $x, 0 \leq x \leq p-2$ tale che

$$\alpha^x \equiv \beta \mod p \tag{9.4}$$

Useremo per il logaritmo finito x la notazione $\log_\alpha \beta$

Prendiamo il caso di $\boldsymbol{Z_{11}}$ con $\alpha = 7$ primitivo. Nella tabella 9.2 è riportato il valore del logaritmo in base 7 di β. Naturalmente quando p diventa molto grande

β	1	2	3	4	5	6	7	8	9	10
$x = \log_7 \beta$	0	3	4	6	2	7	1	9	8	5

Tabella 9.2 Tavola dei logaritmi in $\boldsymbol{Z_{11}}$.

la consultazione di una tabella simile risulta di fatto inattuabile.

L'asimmetria sottesa dal problema del logaritmo discreto tra funzione diretta (elevamento a potenza) e funzione inversa (calcolo del logaritmo) è resa evidente da alcune valutazioni euristiche che stimano la complessità dei migliori algoritmi disponibili. L'elevamento a potenza si può infatti realizzare mediante successive riduzioni modulo p di potenze parziali; p.es. $\alpha^{12} \mod p$ può essere scritto come

$$\left(\left(\alpha^2 \mod p\right)^2 \mod p\right)^3 \mod p \tag{9.5}$$

Per l'esponenziazione esistono dunque algoritmi con ordine di complessità pari a $O(m)$ (con m il numero di bit necessari per rappresentare p), mentre per il calcolo del logaritmo i migliori algoritmi disponibili hanno un ordine di grandezza pari a $O(e^{a\sqrt{m \ln m}})$ (a è un'opportuna costante ≥ 1). Anche se la crescita di quest'ultimo ordine d'infinito è sub-esponenziale, per m abbastanza grande ci si

cautela ampiamente. Per esempio con $p = 2^{512}$ e un tasso pari a 10^9 operazioni al secondo volendo calcolare il logaritmo si tiene occupato il calcolatore per oltre 2120 anni, mentre bastano 5.1210^{-7} secondi per l'elevamento a potenza.

Nel citato lavoro di Diffie e Hellman si usa un problema di stretta derivazione del *PLD* (il *PDH*), che per l'importanza assunta formuliamo e denominiamo a parte.

PDH Assegnati p primo, α elemento primitivo di $\boldsymbol{Z_p}$ e i due elementi $\alpha^x \mod p$ e $\alpha^y \mod p$, si trovi $\alpha^{xy} \mod p$.

È noto che *PDH* è riducibile a *PLD* in tempo polinomiale, anche se non è stata stabilita l'equivalenza computazionale. Per un'analisi degli algoritmi più importanti e per la valutazione del loro campo di maggior efficacia rimandiamo a [82], [86] e [64].

9.2.3 Scomposizione in fattori primi

Il problema della scomposizione di un numero nei suoi fattori primi è sicuramente il più importante tra quelli che analizzeremo. Ciò è dovuto in gran parte al fatto che su di esso si basa il cifrario *RSA*, che ha acquisito molta popolarità grazie alla robustezza che ha di fatto dimostrato. Nonostante risalga al 1978, il cifrario concepito da Rivest, Shamir e Adleman ha sostanzialmente resistito agli attacchi di cui è stato oggetto nel corso degli ultimi vent'anni.

Da segnalare inoltre l'impiego di questa funzione unidirezionale anche nello schema a chiave pubblica di Rabin, di notevole interesse teorico-normativo, e nel campo delle firme numeriche con uno schema basato ancora una volta sul sistema *RSA*.

Il problema della *scomposizione in fattori primi* (*SFP*) ha la seguente formulazione

SFP Assegnato un numero intero n, trovare la sua scomposizione in fattori primi

$$n = p_1^{n_1} p_2^{n_2} \dots p_r^{n_r} \tag{9.6}$$

con p_i fattori primi e $n_i \geq 1$.

Per il problema *SFP* sono disponibili diversi algoritmi; molti di questi hanno prestazioni più spinte quando la fattorizzazione ricade in una qualche classe particolare, p.es. se i fattori p_i sono tutti piccoli oppure fortemente disomogenei. Il modo migliore per tentare la scomposizione è forse quello di usare una strategia mista, basata sull'applicazione concatena dei diversi algoritmi noti, e per questo rimandiamo il lettore interessato alla bibliografia di [64]. In ogni caso si stima

che la complessità computazionale dei migliori algoritmi disponibili abbia un ordine pari a $O(a^{\sqrt{m \ln m}})$ (con $a > 1$ ed m numero di bit per la rappresentazione di n). Si osservi inoltre che il corrispondente problema di decisione "Assegnato un numero intero n, trovare se è scomponibile" è considerato computazionalmente più semplice, anche se non c'è al momento evidenza che appartenga a *P*. L'algoritmo di riferimento, dovuto a Jacobi, ha un ordine di grandezza pari a $O(m^{a \ln \ln m})$, che è "poco più " che polinomiale, almeno per i valori di n d'interesse crittografico.

Come vedremo nel seguito, il cifrario *RSA* considera numeri esprimibili nella forma $n = pq$, con p e q fattori primi, e si ricava direttamente dal seguente problema (*PRSA*)

PRSA Siano assegnati un numero intero n, dato dal prodotto di due primi distinti p e q, un intero positivo y, con $(y, (p-1)(q-1)) = 1$ e un intero c; si trovi m tale che

$$m^y \equiv c \mod n \tag{9.7}$$

Si osservi che il *PRSA* si può ridurre in tempo polinomiale al problema *SFP*, ma non è noto se siano computazionalmente equivalenti.

9.2.4 Generazione di numeri primi elevati

Tanto nel problema del logaritmo finito quanto nel *PRSA* si deve far uso di numeri primi. Nel *PDH* (o anche in *PLD*) bisogna infatti costruire $\boldsymbol{Z}_{\boldsymbol{p}}^{*}$ con p primo, mentre nel *PRSA* serve addirittura una coppia di primi, p e q, scelti con una probabilità il più possibile uniforme nell'intervallo considerato. Poiché tutti i sistemi che si basano sulle funzioni unidirezionali lavorano in modo adeguatamente sicuro solo quando i numeri coinvolti sono molto elevati (e ciò per far esplodere il tempo di calcolo necessario nell'inversione della funzione unidirezionale), è ovvio che anche p e q devono essere primi molto grandi. Per molto grandi, in ambito crittografico, s'intendono oramai numeri con molte centinaia di bit, e trovare dei primi del genere non è facile, poiché non sono noti algoritmi efficienti. Ciò è in qualche modo legato al fatto, già ricordato precedentemente, che non sono noti algoritmi polinomiali per accertare la primalità di un intero.

I numeri primi non sono tutto sommato troppo rari, almeno finché si rimane su valori accettabili della lunghezza dell'intervallo ispezionato. Un importante teorema della teoria dei numeri stabilisce che la loro legge di accrescimento $\pi(x)$, dove $\pi(x)$ è il numero di primi $\leq x$, è tale che

$$\lim_{x \to \infty} \frac{\pi(x)}{x / \ln x} = 1 \tag{9.8}$$

Ciò significa che la porzione di primi $\leq x$ è circa pari a $1/\ln x$, che con $x = 2^{1024}$ porta a circa un primo ogni 355 interi dispari (si tenga però conto che essi non sono distribuiti in modo omogeneo).

Per generare primi elevati si adotta in pratica il seguente procedimento: si sceglie in modo casuale un intero dispari con un numero adeguato di bit, e si effettua a posteriori una *verifica di primalità*. Se essa dà esito positivo si accoglie il numero scelto, altrimenti si effettua un'altra verifica usando un altro intero.

Nonostante esistano alcuni algoritmi che forniscono risposte certe in tempo sub-esponenziale (che per l'intervallo di nostro interesse è "quasi" polinomiale), in pratica si ricorre quasi sempre ad *algoritmi di tipo probabilistico*. Loro caratteristica comune è data dal fatto che forniscono delle risposte affette da una certa probabilità ϵ di errore, che può però essere resa arbitrariamente piccola (algoritmi probabilistici di tipo *Monte-Carlo*), oppure rispondono solo con probabilità ϵ, ma fornendo soluzioni esatte (algoritmi probabilistici di tipo *Las Vegas*).

Nel nostro caso si ha un algoritmo di tipo *Monte-Carlo*, e gli interi dichiarati "fattorizzabili" lo sono con certezza assoluta; tuttavia esiste una piccola probabilità che un intero dichiarato "primo" non lo sia effettivamente. Questa probabilità si può controllare iterando la verifica di un certo predicato un numero adeguato di volte. Per questo motivo tali interi sono chiamati anche *pseudoprimi*.

In questa sezione eviteremo di parlare degli algoritmi tradizionali (poco usati in ambito crittografico), ma ci concentreremo sugli algoritmi probabilistici di *Solovay-Strassen* e di *Miller-Rabin*.

In entrambi i casi si sceglie un intero dispari n a caso, e successivamente k interi i compresi tra 1 e $n-1$. Si dispone inoltre di un *predicato* $\mathfrak{P}(i,n)$ la cui validità viene verificata per tutti i k interi i. La struttura generale della verifica è la seguente

Verifica di primalità

1. se n è primo, il predicato $\mathfrak{P}(i,n)$ *vale* per tutti i k interi;
2. se n è scomponibile, il predicato $\mathfrak{P}(i,n)$ *non vale* per almeno una certa frazione α dei k interi.

Per descrivere materialmente i due algoritmi dobbiamo introdurre le funzioni aritmetiche di *Legendre* e di *Jacobi*. Cominciamo col definire il *resto quadratico modulo* p.

Def. 9.4. *Se p è un primo dispari e a è un qualunque intero compreso tra 1 e $p-1$, si dice che a è un resto quadratico modulo p se esiste $y \in \boldsymbol{Z}_p^*$ tale che*

$$y^2 \equiv a \mod p \tag{9.9}$$

Se chiamiamo Q_p l'insieme dei resti quadratici mod p, il *Simbolo di Legendre* $\mathcal{L}(a,p)$ $(a \geq 0)$ si definisce nel modo seguente

$$\mathcal{L}(a,p) = \begin{cases} 0 & \text{se } a \equiv 0 \mod p \\ 1 & \text{se } a \in Q_p \\ -1 & \text{se } a \notin Q_p \end{cases} \tag{9.10}$$

Prendendo invece un intero dispari scomponibile $n = p_1^{n_1} p_2^{n_2} \dots p_r^{n_r}$ e un qualunque altro intero a, si può definire il *Simbolo di Jacobi*, $\mathcal{J}(a,n)$, che appare come una sorta di generalizzazione del simbolo di Legendre

$$\mathcal{J}(a,n) = \prod_{i=1}^{r} \mathcal{L}(a,p_i)^{n_i} \tag{9.11}$$

La verifica di primalità di Solovay-Strassen si basa essenzialmente sui seguenti teoremi della teoria dei numeri, per la dimostrazione dei quali rimandiamo il lettore a [50]

Teorema 9.1. *Sia p un primo dispari.*

1. Un intero a è un resto quadratico mod p se e solo se

$$a^{(p-1)/2} \equiv 1 \mod p \tag{9.12}$$

2. (Criterio di Eulero) Per qualunque a si ha

$$\mathcal{L}(a,p) \equiv a^{(p-1)/2} \mod p \tag{9.13}$$

Il teorema mostra che, prendendo un qualunque intero n, se questo è primo si ha sicuramente $\mathcal{J}(a,n) \equiv a^{(n-1)/2} \mod n$ per qualunque a. Se viceversa n è fattorizzabile non si sa a priori se la relazione di equivalenza valga o meno; essa vale solo per i cosiddetti *pseudoprimi di Eulero di base a*, dove a viene definito *mentitore* di Eulero. Poiché si può dimostrare che i mentitori di Eulero per n sono al più $\Phi(n)/2 < n/2$, la relazione (9.13) vale per al più metà degli interi $1 \leq a \leq n-1$.

L'algoritmo complessivo per la verifica della primalità si descrive nel modo seguente

Verifica di primalità di Solovay-Strassen

1. Fissato n si scelga un intero a caso $a, 1 \leq a \leq n-1$.
2. **If** $\mathcal{J}(a,n) \equiv a^{(n-1)/2} \mod n$ **then** "n è primo"

   else "n è fattorizzabile"

Oss. 9.5. Il calcolo del simbolo di Jacobi basato sulla sua definizione presuppone la conoscenza della scomposizione di n, introducendo un circolo vizioso. Il problema si risolve poiché esiste il seguente metodo ricorsivo per il calcolo efficiente di $\mathcal{J}(a,n)$

Calcolo ricorsivo del simbolo di Jacobi

1. $\mathcal{J}(1,n)=1$
2. **If** a è pari **then** $\mathcal{J}(a,n)=\mathcal{J}(a/2,n)(-1)^{(n^2-1)/8}$

   else $\mathcal{J}(a,n)=\mathcal{J}(n \mod a,a)(-1)^{(a-1)(n-1)/4}$

Avendo scelto un solo intero a per effettuare la verifica, la probabilità di errore nel caso di numero composto è limitata dal valore $1/2$. Se ripetiamo la verifica usando k interi distinti e supposto di aver avuto sempre indicazione che n è primo, la probabilità d'errore è dell'ordine di $1/2^k$ (per una sua valutazione precisa si veda [82]). Con $k=100$ abbiamo errore dell'ordine di 2^{-100}, che è assolutamente trascurabile, e che ci induce a porre la massima fiducia sul fatto che n sia effettivamente primo.

L'algoritmo di *Solovay-Strassen*, introdotto nel 1977, è stato usato con profitto per molti anni. Attualmente si tende però a impiegare l'algoritmo proposto da *Miller* e *Rabin*; nonostante una sua maggior complessità concettuale e descrittiva, esso è meno pesante dal punto di vista computazionale; non è inoltre richiesto il calcolo del simbolo di Jacobi. Un altro vantaggio concreto è dato dal fatto che, mentre nel caso dell'algoritmo di *Solovay-Strassen* la frazione di interi a che superano il predicato con n composto è al più 1/2, nel caso dell'algoritmo di *Miller-Rabin* questa frazione scende a 1/4, in quanto i pseudoprimi che soddisfano il predicato della verifica di *Miller-Rabin*, detti anche *pseudoprimi forti*, sono meno di 1/4 degli interi compresi tra 1 e $n-1$. Ciò consente un numero minore di verifiche a parità di probabilità d'errore. Anche in questo caso, se il numero è effettivamente primo viene riconosciuto come tale con certezza assoluta.

Verifica di primalità di Miller-Rabin

1. Fissato n (dispari) si scelga come r il più alto esponente di 2 che divida $n-1$, cioè
$$n-1=2^r m$$
2. si scelga un intero a caso a, $1 \leq a \leq n-1$;
3. si calcoli $b=a^m \mod n$;

4. **If** $b \equiv 1 \mod n$ **then** "n è primo" **end**;

5. **for** $i := 0$ **to** $r - 1$ **do**
   If $b \equiv -1 \mod n$ **then** "n è primo" **end**
   else $b \leftarrow b^2 \mod n$

6. "n è scomponibile" **end**

Per l'analisi di questi algoritmi di verifica della primalità rimandiamo ai testi più specialistici [50, 82, 72].

9.2.5 Il problema delle radici quadrate in Z_n

Si consideri l'equazione di secondo grado in $\boldsymbol{Z_n^*}$

$$x^2 + \beta x \equiv c \mod n \tag{9.14}$$

nell'incognita x. Effettuando un cambio di variabili opportuno, $x = y - 2^{-1}\beta$, si può eliminare il termine lineare, ottenendo

$$y^2 \equiv c + 4^{-1}\beta^2 \mod n \qquad \text{cioè} \qquad y^2 \equiv a \mod n \tag{9.15}$$

con $a = c + 4^{-1}\beta^2$. La soluzione dell'equazione si riconduce al calcolo della radice quadrata di un intero a in $\boldsymbol{Z_n}$ modulo n. Ricordiamo dalla definizione 9.4 che se esiste soluzione all'equazione (9.15), a si dice resto quadratico $\mod p$.

Ebbene accade che la (9.15) sia facilmente risolubile solo nel caso in cui si abbia $n = p$ con p numero primo. Più precisamente possiamo dire che in tal caso è noto un algoritmo probabilistico efficiente con una complessità dell'ordine di $O((\log p)^4)$. Tuttavia, se per p vale anche una delle due seguenti condizioni particolari

$$p \equiv 3 \mod 4 \qquad \text{oppure} \qquad p \equiv 5 \mod 8 \tag{9.16}$$

allora la radice quadrata si può calcolare in modo *deterministico*. Per il primo caso, $p \equiv 3 \mod 4$, vale il seguente

Lemma 9.1. *Se p primo è esprimibile nella forma $p = 4k+3 = 4k-1$, allora una soluzione dell'equazione $y^2 \equiv a \mod p$, con a resto quadratico $\mod p$, è data da $y = a^k \mod p$.*

Dim. Si effettua la verifica diretta, scrivendo k nella forma $k = (p+1)/4$

$$a^{(p+1)/4} a^{(p+1)/4} = a^{(p+1)/2} = a^{(p-1)/2} a \equiv a \mod p$$

dove l'ultima uguaglianza deriva dalla (9.12) □

Se si tenta la risoluzione dell'equazione (9.14) con n scomponibile le cose si complicano molto, a meno che *non si conosca la scomposizione di* n. In quest'ultimo caso, nell'ipotesi che sia $n = pq$ con p e q noti, ci si può infatti ridurre alla soluzione di due equazioni del tipo (9.14) modulo p e modulo q

$$\begin{aligned} z_1^2 + \beta z_1 &\equiv c \mod p \\ z_2^2 + \beta z_2 &\equiv c \mod q \end{aligned} \tag{9.17}$$

dove le soluzioni alla (9.14) sono esprimibili nella forma

$$x = az_1 + bz_2 \tag{9.18}$$

con

$$a \equiv \begin{cases} 1 \mod p \\ 0 \mod q \end{cases} \qquad b \equiv \begin{cases} 0 \mod p \\ 1 \mod q \end{cases} \tag{9.19}$$

che deriva dall'applicazione del *Teorema del resto cinese* (vedi [50]).
Se non è nota la scomposizione di n, allora siamo di fronte al seguente *problema della radice quadrata in* $\boldsymbol{Z_n}$ (*PRQZ*)

PRQZ. Assegnati un numero intero n, dato dal prodotto di due primi distinti p e q, e un intero a resto quadratico $\mod n$ (cioè $a \in Q_n$), si trovi la radice quadrata di $a \mod n$.

Sulla difficoltà di trovare le soluzioni del problema *PRQZ* si basano alcune tecniche crittografiche, la più importante delle quali è il cifrario a chiave pubblica di *Rabin*, che analizzeremo nella sezione 9.3.3.

9.3 Cifrari a chiave pubblica

Passeremo ora brevemente in rassegna alcuni tra i più importanti cifrari a chiave pubblica associati alle funzioni unidirezionali viste nella sezione 9.2.1. La struttura di alcuni di questi può essere opportunamente impiegata anche nell'ambito del problema dell'autenticazione e delle firme numeriche (vedi capitolo 10).

9.3.1 Cifrario delle somme parziali

Anche se il cifrario delle somme parziali (o *knapsack cipher*) non è più attuale dal punto di vista della sicurezza crittanalitica, è comunque opportuno descriverlo nei dettagli, e ciò non solo per l'innegabile interesse storico, ma anche per l'eleganza formale del metodo. Esso fornisce infatti un esempio molto limpido di cosa s'intenda per funzione unidirezionale ad accesso.

L'idea sul quale si basa il cifrario è la seguente. Il problema *PSP* delle somme parziali, analizzato nel paragrafo 9.2.1, è *NP-completo*; ciò significa che non sono noti algoritmi in tempo polinomiale per risolvere le sue istanze più difficili. Assegnato dunque un insieme di numeri interi $A = \{a_1, a_2, \ldots, a_n\}$ e un intero S, quando n tende all'infinito il miglior algoritmo diponibile in grado di decidere se esiste un sottinsieme di A che somma a S ha una crescita esplosiva del numero di iterazioni. Ricordiamo però che in certi casi esistono istanze speciali che consentono una risoluzione in tempo polinomiale, e l'ipotesi di supercrescenza (9.3) è una di queste. In tabella 9.3 riportiamo l'algoritmo lineare di risoluzione, già abbozzato nel paragrafo 9.2.1. Il problema diretto, facilmente

for $i = n$ **downto** 1 **do**
  if $S \geq a_i$ **then**
    begin
      $S := S - a_i$
      $x_i := 1$
    end
  else $x_i := 0$
if $S = 0$ **then** "$X = \{x_1, x_2, \ldots, x_n\}$ è la soluzione"
else "non ci sono soluzioni"

Tabella 9.3 Algoritmo per risolvere l'istanza supercrescente.

risolubile in tempo polinomiale, è allora quello di calcolare S assegnati gli insiemi $X = \{x_1, x_2, \ldots, x_n\}$ e $A = \{a_1, a_2, \ldots, a_n\}$; ciò potrebbe corrispondere all'operazione di cifratura. Viceversa la decrittazione dovrebbe essere associata alla risoluzione del problema (*NP-completo*) di ricavare $X = \{x_1, x_2, \ldots, x_n\}$ dati S e A. Tuttavia se usiamo un qualunque insieme A anche la decifrazione diventa impossibile. Si effettua allora la decifrazione usando un'ennupla supercrescente A_S, in modo tale che l'utente legittimo riesca a compiere l'inversione in un tempo polinomiale. Per evitare che anche il crittanalista possa usare quest'informazione in fase di crittanalisi, l'informazione pubblica relativa all'ennupla A_S viene "mascherata" attraverso la trasformazione di A_S in un'ennupla non supercrescente, e quindi "difficile" dal punto della complessità computazionale necessaria per l'inversione. Il mascheramento viene realizzato mediante delle riduzioni modulo un opportuno intero p. Lo schema complessivo dell'algoritmo è il seguente

Cifrario delle somme parziali

1. Si scelgono:
   (a) un intero n;

(b) un'ennupla supercrescente $A_S = \{a_1, a_2, \ldots, a_n\}$;

(c) un intero $u : u > 2a_n$;

(d) un intero $v : v < u$ e $(u, v) = 1$;

2. si calcola $v^{-1} : \ v^{-1}v \equiv 1 \mod u$;

3. la chiave (privata) di decifrazione è data da A_S, u, v^{-1};

4. la chiave (pubblica) di cifratura è data da

$$D = \{d_1, d_2, \ldots, d_n\} = vA_S \mod u$$

5. dato il messaggio m sotto forma di ennupla binaria, la *cifratura* è data da

$$c = D \cdot m = \sum_{i=1}^{n} d_i m_i$$

6. ricevuto il crittogramma c, per eseguire la *decifrazione* si deve effettuare l'operazione

$$c^* = v^{-1}c \mod u$$

La decifrazione consente di recuperare il messaggio m risolvendo l'istanza supercrescente (facile) associata all'informazione segreta A_S in luogo di quella difficile basata sull'informazione pubblica D. Infatti

$$c^* = v^{-1}c \mod u = v^{-1}D \cdot m \mod u = v^{-1}v\, A_S \cdot m \mod u = \\ = A_S \cdot m \mod u = A_S \cdot m \tag{9.20}$$

Dati dunque c^* e A_S si ricava m in tempo polinomiale. Si osservi che l'ultima uguaglianza deriva dalla 1.(c) dell'algoritmo e dalla supercrescenza della A_S, che implica $A_S \cdot m < 2a_n < u$.

Nel linguaggio delle funzioni unidirezionali ad accesso possiamo dire che la risoluzione del problema inverso delle somme parziali è inattuabile se tentata attraverso l'ennupla pubblica D; viceversa la conoscenza dell'informazione suppletiva A_S consente una facile inversione.

Una breve nota sul punto 2 dell'algoritmo. In esso si presuppone il calcolo di v^{-1}, reciproco di v modulo p. Tale operazione si effettua facilmente sulla base del seguente *teorema di Eulero*

Teorema 9.2 (di Eulero). *Assegnato un qualunque intero* v $(1 \leq v \leq u)$ *con* $(v, u) = 1$ *vale la relazione*

$$v^{\Phi(u)} \equiv 1 \mod u \tag{9.21}$$

dove $\Phi(.)$ è la funzione di Eulero (8.13) già definita precedentemente. Si noti che nel caso in cui $u = p$ sia primo, allora $\Phi(p) = p - 1$ e il corollario che se ne ricava, cioé $v^{p-1} \equiv 1 \mod p$, è noto come *teorema di Fermat* . Entrambi i teoremi derivano direttamente dal teorema di *Lagrange* della teoria dei gruppi, che stabilisce che l'ordine di un elemento α di un gruppo moltiplicativo G di ordine n divide n [18].

Per trovare il reciproco di v basta esprimere il teorema nella forma $v^{\Phi(u)} \equiv v^{-1}v \mod u$, ottenendo

$$v^{-1} \equiv v^{\Phi(u)-1} \mod u \tag{9.22}$$

Esempio 9.1. Prendiamo come ennupla supercrescente la

$$A_S = \{3, 8, 13, 27, 51\}$$

scegliendo $u = 132 = 2^2 \cdot 3 \cdot 11$, $v = 19$ coprimo con 132. Calcoliamo ora l'inverso di 19 modulo 132 usando la (9.22). Per la $\Phi(132)$ si ottiene

$$\Phi(132) = 132\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{11}\right) = 40$$

Si ottiene dunque

$$v^{-1} = v^{\Phi(132)-1} \mod 132 = 19^{39} \mod 132 = 7$$

dove l'ultima operazione viene eseguita riducendo modulo 132 le successive potenze parziali secondo lo schema visto nella (9.5). Nel nostro caso possiamo scrivere

$$\begin{aligned} 19^{39} \mod 132 &= \left(19^3 \mod 132\right)^{13} \mod 132 = 127^{13} \mod 132 = \\ &= \left(127^3 \mod 132\right)^4 127 \mod 132 = \\ &= 7^4 127 \mod 132 = 25 \cdot 127 \mod 132 = 7 \end{aligned}$$

La chiave pubblica di cifratura è

$$D = 19 \cdot \{3, 8, 13, 27, 51\} \mod 132 = \{57, 20, 115, 117, 45\}$$

Se $m = \{0, 1, 1, 0, 1\}$ è il messaggio da trasmettere, la cifratura porta al crittogramma

$$c = \{0, 1, 1, 0, 1\} \cdot \{57, 20, 115, 117, 45\} = 180$$

Ricevuto il crittogramma la decifrazione si effettua moltiplicando lo stesso per $v^{-1} = 7$ modulo 132:

$$c^* = 7 \cdot 180 \mod 132 = 72$$

Risolvendo l'istanza supercrescente

$$72 = m \cdot \{3, 8, 13, 27, 51\}$$

secondo l'algoritmo della tabella 9.3 si riottiene $m = \{0, 1, 1, 0, 1\}$ ◯

La forzatura del cifrario (e di una sua generalizzazione proposta in un tempo successivo) si basa sull'impiego di un algoritmo di programmazione intera di Lenstra, che consente di individuare in tempo polinomiale una coppia di interi v' e u' tali che v'/u' sia molto prossimo a v/u, e tali che $a'_i = v'd_i \mod u$ sia un'ennupla supercrescente [76].

Vale la pena ricordare anche il cifrario di *Chor-Rivest* [17], che generalizza quello di *Merkle-Hellman*. Esso coniuga l'impiego di elementi di un campo finito $GF(p^k)$, espressi mediante la loro rappresentazione come polinomi su $GF(p)$ di grado $\leq k$, e di operazioni di logaritmo finito. Oltre all'ingombro dei suoi parametri e alla lentezza delle operazioni di decifrazione il cifrario è stato forzato recentemente.

9.3.2 Cifrario RSA

Veniamo ora al cifrario senza dubbio più studiato e più usato tra quelli a chiave pubblica. Esso venne presentato da Rivest, Shamir e Adleman (da cui l'acrostico) nello stesso anno del cifrario di Merkle-Hellman, il 1978; a differenza di quest'ultimo l'*RSA* è però riuscito a mantenere la sua inviolabilità a tutt'oggi, nonostante a priori potesse sembrare più debole in quanto associato a un problema che non appare essere *NP-completo* (scomposizione in fattori primi).

La sicurezza del cifrario è strettamente legata al problema *PRSA*, analizzato nella sezione 9.2.3, la cui difficoltà deriva dalla non conoscenza di algoritmi polinomiali per la scomposizione di un intero nei suoi fattori primi.

Cifrario RSA

1. Si generano due primi elevati p e q usando le verifiche di primalità di *Solovay-Strassen* o *Miller-Rabin* viste nel paragrafo 9.2.4, producendo
$$n = pq$$
2. si calcola $\Phi(n) = n\left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right) = (p-1)(q-1)$
3. si sceglie x coprimo con $\Phi(n)$ e se ne calcola il reciproco $\mod \Phi(n)$ mediante il teorema di Eulero (9.22); esistenza e unicità sono garantite dal fatto che $(x, \Phi(n)) = 1$
$$xy \equiv 1 \mod (\Phi(n)) \qquad (x, \Phi(n)) = 1$$
4. n e y costituiscono la chiave pubblica;
$\Phi(n), p, q, x$ sono la chiave privata

5. se $m \in \boldsymbol{Z_n}$ è il messaggio, cifratura e decifrazione si eseguono nel modo seguente

$$\begin{array}{ll} \text{Cifratura} & c = m^y \mod n \\ \text{Decifrazione} & m = c^x \mod n \end{array} \tag{9.23}$$

La verifica del funzionamento della decifrazione non è immediata, e richiede qualche calcolo. Si tratta in sostanza di dimostrare che

$$c^x = m^{xy} \equiv m \mod n \tag{9.24}$$

Si osservi che l'uguaglianza (9.24) può essere espressa anche mediante le due uguaglianze modulo p e q di seguito riportate

$$\begin{aligned} m^{xy} &\equiv m \mod p \\ m^{xy} &\equiv m \mod q \end{aligned} \tag{9.25}$$

Infatti, se valgono le (9.25) possiamo scrivere

$$\begin{aligned} m^{xy} &= ap + m \\ m^{xy} &= bq + m \end{aligned}$$

con a e b interi opportuni, cioè $ap = bq$; ma essendo p e q primi l'uguaglianza implica anche $b = dp$ (e $a = cq$) e dunque

$$m^{xy} = bq + m = dpq + m = dn + m = m \mod n$$

d'altra parte se vale la (9.24) possiamo scrivere

$$m^{xy} = \alpha n + m = \alpha pq + m$$

e ciò implica la validità delle (9.25).

Per verificare la decifrazione basta allora dimostrare quest'ultime (p.es. la prima delle due), distinguendo tra il caso in cui p sia fattore di m o meno. Se p divide m il tutto è banalmente verificato. Viceversa, tenendo conto del teorema di (Fermat) Eulero (9.21) $m^{p-1} \equiv 1 \mod p$, possiamo scrivere

$$\begin{aligned} m^{xy} \mod p = m^{1+k\Phi(n)} \mod p = m^{1+k(p-1)(q-1)} = m\left(m^{p-1}\right)^{k(q-1)} \equiv \\ \equiv m \mod p \end{aligned} \tag{9.26}$$

che dimostra la validità dell'operazione di decifrazione.

Oss. 9.6. Nel caso particolare in cui m ed n siano coprimi si ha $m^{\Phi(n)} \equiv 1 \mod n$ e la (9.26) si può dimostrare direttamente senza passare per le equivalenze della (9.25). Si tenga però conto che tale ipotesi non è sempre sostenibile.

Esempio 9.2. Prendiamo $p = 211$ e $q = 439$, che superano la verifica di *Miller-Rabin*. Si ottengono i seguenti parametri: $n = 211 \cdot 439 = 92629, \Phi(n) = 210 \cdot 438 = 91980$. Scegliamo ora $x = 4493$, che è primo con $\Phi(n)$, e calcoliamo il reciproco modulo $\Phi(n)$, usando il teorema di Eulero e la formula (9.22). Si ottiene

$$y = 4493^{\Phi(\Phi(n))-1} \mod \Phi(n) = 4493^{20735} \mod 91980 = 81437$$

La chiave pubblica è data dalla coppia $n = 92629$ e $y = 81437$. Se il messaggio è $m = 13455$ il crittogramma che si ottiene è

$$c = 13455^{81437} \mod 92629 = 91870$$

Per decifrarlo bisogna eseguire l'operazione

$$m = c^x \mod 92629 = 91870^{4493} \mod 92629 \equiv 13455 \mod 92629$$

◯

A parte la scomposizione di n in p e q, per la quale non sono noti algoritmi in tempo polinomiale, sono stati tentati numerosi altri attacchi al cifrario; il più ovvio è quello di recuperare m dal corrispondente c disponendo della chiave pubblica n, y; ciò corrisponde al problema *PRSA* già anticipato nella sezione 9.2.3, per il quale non sono noti algoritmi efficienti. D'altra parte anche l'ipotesi di ricavare $\Phi(n)$ da n non regge. Essa consentirebbe infatti l'immediata scomposizione di n, per la quale abbiamo appena detto non esistono algoritmi efficienti. Infatti da

$$\begin{aligned} \Phi(n) &= (p-1)(q-1) = n + 1 - (p+q) \\ (p-q)^2 &= (p+q)^2 - 4n \end{aligned}$$

si ricaverebbero $p+q$ e $p-q$ in funzione di fattori noti, cioè

$$\begin{aligned} p + q &= n - \Phi(n) + 1 \\ p - q &= \pm\sqrt{(p+q)^2 - 4n} \end{aligned}$$

il che porta all'individuazione di p e q.

Un aspetto importante stabilito dalla teoria è che qualunque algoritmo per il calcolo dell'esponente x della decifrazione può essere usato come sotto programma per un algoritmo probabilistico di tipo *Las Vegas* (cfr. 9.2.4) che realizza la scomposizione di n. Poiché un algoritmo di questo tipo fornisce una risposta con una certa probabilità ϵ, ma se risponde lo fa correttamente, effettuando un numero idoneo di iterazioni si può giungere a una risposta esatta con una probabilità arbitrariamente prossima a 1. Su [64] si trova un'ampia rassegna dei possibili attacchi al cifrario *RSA*.

Poiché per effettuare la forzatura del cifrario non sono state trovate finora strade alternative alla scomposizione di n, si è portati a pensare che la complessità della forzatura sia equivalente alla complessità della fattorizzazione di un intero, anche se in studi recenti [12] si è cominciato anche a mettere in dubbio quest'ipotesi.

Nel paragrafo successivo analizzeremo invece un cifrario per il quale tale equivalenza è stata stabilita in modo rigoroso.

Concludiamo il paragrafo ricordando che uno dei maggiori svantaggi del sistema *RSA* (e dei cifrari a chiave pubblica in generale) è che per disporre di un livello adeguato di sicurezza, i numeri coinvolti nella costruzione delle chiavi devono avere dimensioni rilevanti. In generale si tende oramai a impiegare valori dell'ordine del migliaio di bit per il modulo n. Ciò appesantisce notevolmente le operazioni di cifratura e decifrazione, limitando la diffusione e i campi d'impiego del sistema *RSA*, soprattutto se si effettuano confronti con cifrari a chiave segreta quali il *DES*, che consentono di smaltire ingenti masse di dati in brevissimo tempo.

9.3.3 Cifrario di Rabin

Il cifrario di *Rabin* si basa sul problema *PRQZ* della radice quadrata in $\boldsymbol{Z_n}$, analizzato nel paragrafo 9.2.5. Esso riveste un'importanza teorica notevole, poiché al contrario del sistema *RSA* per esso è stato possibile provare che la complessità computazionale associata alla forzatura è *equivalente* alla complessità computazionale relativa alla scomposizione di un intero n. Nel paragrafo 9.2.5 si è visto che si può risolvere facilmente il *PRQZ* a patto di conoscere la scomposizione di n; si può però dimostrare che vale anche il viceversa, e dunque la risoluzione del *PRQZ* comporta la conoscenza di un algoritmo polinomiale per la scomposizione di n.

Passiamo alla descrizione del cifrario.

Cifrario di Rabin

1. Si generano due primi elevati p e q ($p, q \equiv 3 \mod 4$) usando le verifiche di primalità di *Solovay-Strassen* o *Miller-Rabin* viste nel paragrafo 9.2.4, producendo
$$n = pq$$
2. Si sceglie $\beta < n$.
3. La *chiave pubblica* è data da n e β;
la *chiave privata* è data dalla fattorizzazione p, q di n.

4. Se m è un messaggio ($1 \leq m \leq n-1$) la *cifratura* corrisponde all'operazione

$$c = m(m+\beta) \mod n \tag{9.27}$$

5. La *decifrazione* richiede la soluzione dell'equazione (9.15)

$$y^2 = c + 4^{-1}\beta^2 \mod n \tag{9.28}$$

A seguito di quanto visto nel paragrafo 9.2.5, noti p e q l'equazione di decifrazione si risolve in tempo polinomiale mediante le (9.17). Ciò porta all'individuazione di 4 soluzioni, e l'utente destinatario del crittogramma deve decidere quale di queste corrisponde al messaggio inviato. Ciò costituisce un problema solo in linea di principio, poiché l'introduzione di opportuna ridondanza nel messaggio consente di discriminare quello corretto.

Esempio 9.3. Si prenda $p = 13$ e $q = 29$, cioè $n = 377$. Si scelga inoltre $\beta = 115$. La cifratura porta a

$$c = 34(34+115) \mod 377 = 165 \tag{9.29}$$

Le due equazioni da risolvere per la decifrazione sono rispettivamente

$$\begin{aligned} z_1^2 + 115z_1 &= 165 \mod 13 \\ z_2^2 + 115z_2 &= 165 \mod 29 \end{aligned} \tag{9.30}$$

che risolte portano alle soluzioni $z_{11} = 7, z_{12} = 8$, e $z_{21} = 5, z_{22} = 25$. I valori di a e b che soddisfano la (9.19) sono rispettivamente $a = 261$ e $b = 117$. Le quattro soluzioni per l'equazione (9.29) si ricavano dalle seguenti uguaglianze

$$x = \begin{cases} 7 \cdot 261 + 5 \cdot 117 & \equiv 150 & \mod 377 \\ 7 \cdot 261 + 25 \cdot 117 & \equiv 228 & \mod 377 \\ 8 \cdot 261 + 5 \cdot 117 & \equiv 34 & \mod 377 \\ 8 \cdot 261 + 25 \cdot 117 & \equiv 112 & \mod 377 \end{cases}$$

La scelta $m = 34$ va fatta sulla base della ridondanza relativa al messaggio.◯

Per quanto riguarda la sicurezza del cifrario possiamo affermare che, a seguito delle osservazioni fatte sull'equivalenza tra complessità di *PRQZ* e di *SFP*, il cifrario è sicuro nella misura in cui si ritenga intrattabile il problema della scomposizione. Si deve inoltre notare che, nonostante il sistema sia sicuro nei confronti di attacchi crittanalitici con testo in chiaro, esiste la possibilità di attuare con successo *attacchi crittanalitici con testo cifrato*, usando un algoritmo probabilistico tipo *Las Vegas* che fattorizza n con probabilità 1/2. L'introduzione della ridondanza di cui si è accennato precedentemente consente però di cautelarsi entro certi limiti da quest'ipotesi.

9.3.4 Cifrario di ElGamal

Il cifrario ideato da ElGamal è un primo esempio di *cifrario stocastico*, nel quale accanto alle informazioni legate alla chiave e ai parametri scelti il mittente introduce nel crittogramma un numero scelto su base aleatoria. Questo fatto implica che la cifratura ripetuta di uno stesso messaggio m porta di norma a due crittogrammi diversi. La sicurezza del cifrario si basa sulla presunta intrattabilità del problema *PLD* del logaritmo discreto, che viene usato proficuamente anche nel protocollo di Diffie-Hellman per lo scambio di chiave segrete tra due utenti senza che questi abbiano la necessità d'incontrarsi (si veda 10.1.2).

Per attuare il cifrario si sceglie un primo p molto grande, generato mediante gli algoritmi probabilistici di Miller-Rabin o di Solovay-Strassen; si sceglie inoltre una radice primitiva α di $\boldsymbol{Z}_{\boldsymbol{p}}^{*}$, con la quale si possono generare tutti gli elementi $\alpha^0 = 1, \alpha^1, \alpha^2, \ldots, \alpha^{p-2}$ di $\boldsymbol{Z}_{\boldsymbol{p}}^{*}$ (ricordiamo che $\alpha^{p-1} = 1 \mod p$). Gli elementi p e α sono comuni a tutti gli utenti e fanno parte integrante delle chiavi pubblicate. Ecco la struttura dell'algoritmo.

Cifrario di ElGamal

1. L'utente X sceglie un esponente $0 \leq x \leq p-2$ e calcola

   $$\alpha^x \mod p$$

   che viene pubblicato.

2. La *chiave pubblica* di X è data da p, α, α^x;
   la *chiave privata* di X è data da x.

3. Se Y vuole mandare un messaggio m $(0 \leq m \leq p-1)$ a X seleziona un intero k in modo aleatorio $(1 \leq k \leq p-2)$. Poi calcola

   $$c_1 = \alpha^k \mod p \qquad \text{e} \qquad c_2 = m\,(\alpha^x)^k \mod p = m\,(\alpha^x)^k \mod p$$

   trasmettendo il crittogramma

   $$c = c_1 c_2$$

4. La decifrazione avviene calcolando

   $$m = c_1^{p-1-x} c_2 \mod p$$

La decifrazione consente di recuperare il messaggio in quanto

$$\begin{aligned} c_1^{p-1-x} c_2 &= \left(\alpha^k\right)^{p-1-x} m\,(\alpha^x)^k = \\ &= \left(\alpha^{p-1}\right)^k \alpha^{-kx} m \alpha^{kx} = m \end{aligned}$$

Esempio 9.4. Prendiamo $p = 163$ con $\alpha = 7$ primitivo di $\boldsymbol{Z}^*_{\mathbf{163}}$. L'utente X sceglie $x = 93$ come chiave privata, e pubblica $\alpha^x = 7^{93} \equiv 98 \mod 163$. l'utente Y, che vuole spedire il messaggio $m = 26$, sceglie in modo aleatorio un intero $k = 52$ e calcola

$$\begin{aligned} c_1 &= 7^{52} \equiv 62 \mod 163 \\ c_2 &= 26 \cdot 98^{52} \equiv 161 \mod 163 \end{aligned} \qquad \Longrightarrow \qquad c = (62, 161)$$

Quando X riceve $(62, 161)$ esegue l'operazione

$$m = 62^{163-1-93}121 \mod 163 = 26$$

riottenendo il messaggio spedito. ◯

9.3.5 Il cifrario di Mc Eliece

Analizzeremo ora un'applicazione alla crittografia delle tecniche usate nell'ambito dei codici correttori d'errore. Lo schema del cifrario si basa sulla seguente osservazione: in generale il problema di individuare una decodifica per un codice lineare *arbitrario* è noto per essere *NP-completo*; tuttavia per certi codici lineari particolari esistono algoritmi molto efficienti per attuare la decodifica (vedi capitolo 6). Si può allora attuare un sistema nel quale si trasforma un codice lineare con decodifica efficiente (chiave segreta) in un codice lineare con decodifica intrattabile (chiave pubblica). In questo modo la decodifica efficiente sta alla base dell'operazione di decifrazione, mentre la decodifica "difficile" rimane associata alla decrittazione. Lo schema di base è dunque simile a quello usato per il cifrario delle somme parziali. In generale, dal punto di vista della complessità computazionale, una decodifica a distanza minima (si veda il par.6.2.2) impone il controllo della distanza tra il vettore ricevuto e i q^k vettori del codice. Anche ricorrendo a una decodifica con sindrome bisogna confrontare la stessa con quella relativa al vettore nullo (che è nulla), con quella dei $\binom{n}{1}$ vettori d'errore di peso unitario, con i $\binom{n}{2}$ vettori d'errore di peso 2 ecc., ottenendo un numero di controlli pari a

$$\sum_{i=0}^{t} \binom{n}{i}$$

che per un valore di t non limitato portano comunque a una complessità non polinomiale. Viceversa per certe classi di codici lineari sono noti algoritmi molto efficienti di decodifica in tempo polinomiale (p.es. i codici lineari visti nel capitolo 6).

La struttura del cifrario è la seguente. Si parte con la scelta di una matrice generatrice $\boldsymbol{G}$ di tipo $k \times n$ associata a un codice t-correttore con decodifica

efficiente. McEliece scelse i codici di *Goppa*, che sono codici con parametri $n = 2^m, k = n - mt, d \geq 2t + 1$. La matrice G viene poi "mascherata" mediante l'operazione

$$\widehat{\boldsymbol{G}} = \boldsymbol{AGP} \tag{9.31}$$

dove $\boldsymbol{A}$ è una matrice aleatoria (binaria) $k \times k$ non singolare, mentre $\boldsymbol{P}$ è una matrice $n \times n$ di permutazione. Le chiavi coinvolte sono

$$\begin{array}{ll} \text{Chiave pubblica} & \widehat{\boldsymbol{G}}, t \\ \text{Chiave privata} & \boldsymbol{A}, \boldsymbol{G}, \boldsymbol{P} \end{array} \tag{9.32}$$

Cifrario di McEliece

1. Si voglia trasmettere il vettore $\boldsymbol{m}$ di lunghezza k. Si sceglie allora un *vettore d'errore* aleatorio $\boldsymbol{e}$, con $wt(\boldsymbol{e}) \leq t$, e si calcola
$$\boldsymbol{c} = \boldsymbol{m} \cdot \widehat{\boldsymbol{G}} + \boldsymbol{e} \tag{9.33}$$

2. La decifrazione avviene avvalendosi della chiave privata $\boldsymbol{A}, \boldsymbol{P}$ nel modo seguente: si calcola
$$\begin{aligned} \widehat{\boldsymbol{c}} = \boldsymbol{c}\boldsymbol{P}^{-1} = (\boldsymbol{m} \cdot \widehat{\boldsymbol{G}} + \boldsymbol{e})\boldsymbol{P}^{-1} = \\ = \boldsymbol{mAG} + \boldsymbol{e}\boldsymbol{P}^{-1} \end{aligned} \tag{9.34}$$

3. Il vettore ricevuto corrisponde alla codifica di $\boldsymbol{mA}$ mediante la matrice $\boldsymbol{G}$, con la sovrapposizione di un'ennupla d'errore data dalla permutazione di $\boldsymbol{e}$ mediante la matrice $\boldsymbol{P}$. La procedura (efficiente) di decodifica consente di correggere la configurazione d'errore, in quanto per ipotesi $wt(\boldsymbol{e}) \leq t$. All'uscita del procedimento di decodifica si ha il vettore $\boldsymbol{mA}$.

4. post-moltiplicando $\boldsymbol{mA}$ per $\boldsymbol{A^{-1}}$ si riottiene il messaggio m.

Esempio 9.5. Mostriamo un esempio basato sul codice di Hamming $C(7, 4, 3)$ dell'esempio 6.4; esso è in grado di correggere 1 errore, e dunque si ha $t = 1$.

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Scegliamo inoltre le seguenti matrici $\boldsymbol{A}$ e $\boldsymbol{P}$

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \qquad \boldsymbol{P} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

La matrice $\widehat{\boldsymbol{G}} = \boldsymbol{AGP}$ che si ottiene è

$$\widehat{\boldsymbol{G}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Se il messaggio è $\boldsymbol{m} = (0,1,0,0)$ supponiamo di scegliere $\boldsymbol{e} = (0,1,0,0,0,0,0)$ come ennupla d'errore. Il crittogramma da spedire $\boldsymbol{c} = \boldsymbol{m} \cdot \widehat{\boldsymbol{G}} + \boldsymbol{e}$ è pari a

$$\boldsymbol{c} = (0,0,0,1,0,1,0)$$

Per decifrare dobbiamo ricavare $\widehat{\boldsymbol{c}} = \boldsymbol{cP}^{-1}$, ottenendo

$$\widehat{\boldsymbol{c}} = (0,0,0,0,0,1,1)$$

Dalla struttura della matrice $\boldsymbol{G}$ si vede subito che $\widehat{\boldsymbol{c}}$ corrisponde al vettore della prima riga con un errore in posizione 1. Correggendo l'errore e decodificando si ottiene

$$\boldsymbol{mA} = (1,0,0,0)$$

che moltiplicato per

$$\boldsymbol{A^{-1}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

consente di recuperare il messaggio $\boldsymbol{m} = (0,1,0,0)$. ◯

Il cifrario di McEliece ha ricevuto scarsa attenzione dal punto di vista applicativo, poiché la dimensione delle chiavi è molto onerosa. Per stare in completa sicurezza l'Autore suggeriva di lavorare con $n = 1024, m = 10$ e $t = 50$, che porta a un codice di *Goppa* $C(1024, 524, 101)$. La matrice $\boldsymbol{G}$ ha dimensione 524×1024, il blocco del messaggio è costituito da 524 bit, ma il crittogramma ha 1024 bit. Oltre a chiavi molto ingombranti si ha dunque a che fare anche con un'espansione dei dati in transito sul canale. Inoltre il cifrario è stato recentemente forzato sotto condizioni non troppo restrittive [10].

Capitolo 10
Gestione delle chiavi, autenticazione e firme numeriche

10.1 Distribuzione e condivisione delle chiavi

In tutti i sistemi di cifratura analizzati, tanto in quelli simmetrici (o a chiave segreta) quanto in quelli asimmetrici (o a chiave pubblica), si è sempre assunto implicitamente di avere già a disposizione le chiavi per effettuare le operazioni di cifratura/decifrazione. Nell'introduzione alla terza parte abbiamo menzionato la complessità del problema della gestione delle chiavi, tanto dal punto di vista di una loro generazione che da quello della distribuzione tra più utenti; analizzeremo ora il problema in modo più sistematico.

L'analisi deve svolgersi su diversi piani, poiché per ciascun sistema di cifratura, sia esso a chiave pubblica o a chiave segreta, ci sono metodi di scambio e generazione delle chiavi basati tanto su tecniche simmetriche quanto asimmetriche.

In generale si può affermare che, da un punto di vista strettamente operativo, c'è la tendenza a usare sistemi a chiave pubblica per concordare (o trasmettere a distanza) chiavi simmetriche da usare successivamente nel contesto di una cifratura a chiave segreta tradizionale, p.es. il *DES*. Ciò è dovuto alla circostanza che la velocità dei sistemi a chiave segreta è ancora diversi ordini di grandezza maggiore rispetto a quella dei sistemi a chiave pubblica. Di conseguenza è conveniente usare un sistema come il *DES* per smaltire la grande massa dei dati che gli utenti devono scambiarsi, ma si ricorrere preferenzialmente ai più pesanti sistemi a chiave pubblica per concordare o trasmettere a distanza le chiavi, usufruendo pienamente dei vantaggi offerti dalla pubblicazione delle chiavi di cifratura quando si debba gestire una rete con molti utenti.

L'accordo sulle chiavi da usare deve essere realizzato *prima* di iniziare qualunque attività di cifratura/decifrazione, e costituisce dunque il punto di partenza del protocollo che dà l'avvio allo scambio di messaggi segreti (o di verifica delle identità o altro).

Nella letteratura si suole distingue tra *distribuzione* e *condivisione* delle chiavi.

10.1.1 Distribuzione delle chiavi

Nella *distribuzione* delle chiavi uno dei due utenti genera per conto proprio una chiave segreta, che deve poi essere inviata a un secondo utente per la condivisione; ciò deve accadere impedendo che terze persone riescano a intercettare a chiave, che deve quindi rimanere segreta. Per quanto detto precedentemente la procedura di scambio deve essere attuata in linea di principio *senza* condivisione a priori di altre informazioni segrete tra due utenti, anche se in realtà esistono schemi che prevedono l'acquisizione a priori (da un canale sicuro) di una sorta di *chiave a lungo termine*, usata per successive sessioni di generazione di chiavi correnti.

Un esempio molto interessante di protocollo per la distribuzione di chiavi segrete è quello ideato da *Shamir*, le cui linee essenziali sono state anticipate nella (7.2). Come funzione asimmetrica si può usare il logaritmo discreto, previa scelta e pubblicazione del valore di un intero primo p da usare come modulo.

Protocollo di distribuzione di Shamir

1. L'utente X sceglie una chiave segreta k ($1 \leq k \leq p-1$), che deve essere inviata all'utente Y;

2. X e Y scelgono due interi x e y ($\leq p-2$ e coprimi con $p-1$), da usare successivamente come esponenti; mediante il teorema (9.21) di Eulero vengono calcolati i reciproci x^{-1} e y^{-1} modulo $p-1$;

3. (a) X calcola $k^x \mod p$ e lo spedisce a Y
   (b) Y calcola $(k^x)^y \mod p$ e lo spedisce a X
   (c) X calcola $(k^{xy})^{x^{-1}} = k^y \mod p$ e lo spedisce a Y
   (d) Y calcola $(k^y)^{y^{-1}} = k \mod p$ recuperando la chiave k

Il protocollo, che si attua mediante tre cicli di trasmissione, consente la sicurezza nei confronti di un avversario passivo, ma non l'autenticazione, poiché nel punto 3.(b) un falsificatore F potrebbe impersonare Y senza che X se ne avveda. Come già osservato precedentemente il sistema consente una cifratura e una decifrazione senza l'uso di una vera e propria chiave, ma richiede la commutatività della funzione unidirezionale. La sicurezza del protocollo è basata sulla presunta intrattabilità del problema del logaritmo discreto. In un protocollo di distribuzione è implicita una distinzione gerarchica tra chi ha l'autorità per generare (e quindi imporre) le chiavi e chi invece le accetta come utente.

10.1.2 Condivisione delle chiavi

In questo secondo caso il problema è quello di accordarsi a priori su una chiave, condividendo però l'autorità per la generazione della stessa. Il metodo più famoso, descritto nel più volte citato articolo di *Diffie-Hellman* [25], è quello dello scambio delle chiavi mediante logaritmo discreto. Anche in questo caso si sceglie a priori un numero primo elevato, p, e un elemento α primitivo di $\boldsymbol{Z_p^*}$.

Protocollo di condivisione di Diffie-Hellman

1. L'utente X sceglie un esponente x e spedisce a Y (o pubblica) $\alpha^x \mod p$.
2. L'utente Y sceglie un esponente y e spedisce a X (o pubblica) $\alpha^y \mod p$.
3. I due utenti possono ora condividere la chiave $k = \alpha^{xy} \mod p$, che può essere calcolata tanto da Y che da X nei due modi sotto riportati

$$k = \alpha^{xy} \mod p = (\alpha^x)^y \mod p = (\alpha^y)^x \mod p$$

Anche in questo secondo caso manca la possibilità di autenticare l'interlocutore, poiché il frodatore F potrebbe impersonare Y con pieno successo. Il problema si può in parte ovviare predisponendo una *pubblicazione asseverata* delle chiavi di cifratura α^x e α^y, che ha però il difetto di coinvolgere una terza parte, cioè l'*Autorità Garante* che assevera il registro delle chiavi.

10.2 Schemi di autenticazione

Nel capitolo introduttivo si è detto che la crittologia moderna, oltre alla protezione della riservatezza dell'informazione, ha anche altri e più sofisticati obiettivi, che vanno dai problemi di autenticazione e identificazione di un utente, alla non ripudiabilità nella generazione e trasmissione di un documento, alla verifica dell'integrità dei dati trasmessi nei confronti di eventuali attacchi attivi. Dobbiamo inoltre menzionare altri due problemi collaterali, di grandissima importanza pratica, che sono rilevanti soprattutto in contesti e/o ambienti dove le esigenze di sicurezza devono essere elevatissime (si può pensare p.es. ai *caveau* delle grandi banche, a postazioni militari di elevato valore strategico, alle centrali nucleari ecc.).

1. *Protocolli a conoscenza nulla* - Verifica del livello di privilegio (o identificazione) di un utente X basata su un procedimento interattivo che dimostra la conoscenza che l'utente ha di un'informazione segreta x. La verifica deve essere fatta senza svelare alcuna informazione che riguardi x. L'ente (o la persona) che effettua la verifica potrebbe infatti non essere abilitato alla conoscenza di x.

2. *Condivisione a soglia* - Accesso a determinate risorse riservate solo a beneficio di un sottinsieme congiunto di h utenti nel caso in cui sia $h \geq s$, con s valore di soglia; in tal caso si parla di *schemi a soglia*, in quanto il mancato raggiungimento della stessa non consente l'accesso alle risorse.

In questa sezione tratteremo solo gli aspetti generali relativi al problema dell'autenticazione, rimandando a [80] per una trattazione più completa e sistematica.

Negli schemi di crittografia a chiave segreta (o pubblica) si tiene conto solamente del problema dell'intercettazione dell'informazione da parte di un oppositore *passivo*. Per passivo s'intende che l'intercettatore si limita ad acquisire i dati in transito sul canale, ma non attua alcuna *falsificazione* nei confronti dell'utente legittimo che sta ricevendo le informazioni riservate. Le modalità di un attacco di falsificazione da parte di F sono però variegate, e potrebbero riguardare tanto il tentativo di impersonare *in toto* un utente legittimo X, trasmettendo a Y un messaggio completamente falso che Y attribuisce a X, quanto di modificare, anche solo parzialmente, un messaggio vero che X sta trasmettendo a Y.

In ambito tecnico si distingue tra *identificazione* (o *autenticazione*) e *firma numerica*. Nel primo caso si richiede solo la verifica dell'identità dell'utente, che potrebbe non essere legata alla necessità di scambiare informazioni riservate.

Nel caso della firma numerica oltre alla presenza di un messaggio da firmare entra in campo anche il problema della *non ripudiabilità*, nel quale Y deve poter dimostrare a un eventuale giudice (imparziale) che il messaggio legittimamente ricevuto da X è stato effettivamente trasmesso da quest'ultimo, anche se X potrebbe a un certo punto disconoscerlo.

Nel paragrafo 7.1.2 si è visto come uno schema base a chiave pubblica possa essere impiegato come procedura di autenticazione per un utente X (si veda (7.4)) semplicemente invertendo l'ordine con il quale vengono usate le due funzioni diretta e inversa. Una lieve variante, sempre basata su schemi asimmetrici, prevede il seguente

Protocollo di identificazione

1. Y sceglie a caso un intero m, genera $c = C_x(m)$ e lo spedisce a X.
2. Usando la propria funzione di decifrazione, X produce $D_x(c) = D_x C_x(m)$ e lo spedisce a Y.
3. Y procede all'autenticazione di X verificando che sia $D_x(c) = m$

La verifica dell'identità si basa sul fatto che X è l'unico a poter calcolare $D_x(c)$.

Entrambi gli schemi presuppongono però l'asseverazione dell'elenco delle chiavi da parte di un'*Autorità Garante* .

Una procedura simile può essere impiegata anche nel contesto del *controllo degli accessi* in un sistema operativo di un elaboratore. In questi casi l'utente autorizzato X, che chiede la disponibilità di risorse riservate, deve fornire al sistema il proprio nome identificativo e una *password* π, che deve essere verificata dal sistema stesso prima di concedere a X l'accesso. Per far ciò il sistema deve conoscere l'informazione (*utente, password*)=(X, π), ma se questa viene memorizzata direttamente su un file di sistema c'è il pericolo che un frodatore F riesca a leggerla, impersonando successivamente X. Una possibilità è allora quella di introdurre una funzione unidirezionale C_x, con la sua inversa D_x, conservando nella memoria d sistema la coppia $(X, C_x(\pi))$. All'atto della richiesta di accesso l'utente X deve fornire al sistema la coppia (X, π), mentre il sistema si limiterà a calcolare la quantità $C_x(\pi)$, verificando se coincide con quella memorizzata. Così facendo non c'è più la necessità di mantenere segreta l'informazione $(X, C_x(\pi))$, poiché un ipotetico frodatore dovrebbe ricavare π da $C_x(\pi)$, che per ipotesi di funzione unidirezionale è un problema intrattabile.

10.2.1 Protocolli di verifica a conoscenza nulla

Uno svantaggio del metodo descritto nella (7.4) è dato dal fatto che l'informazione usata per ottenere l'autenticazione potrebbe essere reimpiegata in un tempo successivo da parte di terzi utenti che intendessero impersonare X. Da questo punto di vista è opportuno associare all'informazione contenuta nella verifica alcuni caratteri dipendenti dal contesto, in modo tale da rendere inutili le informazioni relative alle autenticazioni già effettuate. Il protocollo di identificazione appena analizzato consente un primo livello di protezione, poiché è lo stesso Y a generare l'informazione m, che può dunque essere cambiata nel corso del tempo.

In entrambi i casi si ha però un impiego diretto delle funzioni C_x e D_x, che vengono usate per *provare* l'identità di X. X è infatti l'unico utente a conoscere D_x, ed è costretto a sfruttare *in toto* quest'informazione per poter convincere Y della sua identità.

Esiste però la possibilità di realizzare protocolli più raffinati, di tipo interattivo, che consentono di evitare l'impiego diretto dell'informazione che identifica X. In un protocollo *di verifica a conoscenza nulla* X deve convincere Y di possedere un'informazione segreta s, a lui riservata e che lo identifica univocamente, senza dare a Y alcun elemento sull'informazione stessa. L'informazione s viene associata a X per mezzo di un'informazione pubblica, e la dimostrazione della conoscenza di s garantisce l'identità di X.

Più in generale un protocollo di questo genere consente di *dimostrare* la verità di un'affermazione senza fornire alcuna informazione sulla dimostrazione stessa.

Nel protocollo che analizzeremo, dovuto a *Fiat* e *Shamir*, la sicurezza del metodo (cioè la capacità di mantenere celata l'informazione s) si basa sulla difficoltà di effettuare il calcolo di radici quadrate modulo un intero n composito (si veda il problema *PRQZ* 9.2.5). Per il funzionamento del protocollo è necessario che un'*Autorità Garante* (*AG*) generi due primi elevati, p e q, pubblicando $n = pq$, ma conservando segreti i due fattori.

Protocollo di Fiat-Shamir

1. L'utente X sceglie (o conosce) $s : (s, n) = 1, 1 \leq s \leq n - 1$; calcola

$$x = s^2 \mod n$$

   che viene poi certificato da *AG* e pubblicato.

2. I passi seguenti vengono ripetuti un numero t di volte e Y accetta la dimostrazione a conoscenza nulla se il seguente punto (d) viene verificato sempre correttamente.

   (a) X sceglie un numero r a caso ($1 \leq r \leq n - 1$) e spedisce a Y

$$c = r^2 \mod n$$

   (b) Y sceglie a caso un bit di *asseverazione* $b \in \{0, 1\}$, e lo spedisce a X.

   (c) X calcola il *responso* $y = rs^b \mod n$ e lo spedisce a Y; se $b = 0$ si ha $y = r$ mentre se $b = 1$ risulta $y = rs \mod n$.

   (d) Y verifica che sia

$$y^2 \equiv cx^b \mod n$$

   respingendo la dimostrazione se la verifica dà esito negativo.

Un falsificatore F potrebbe impersonare X solo nel caso in cui sia $b = 0$, poiché F risponderebbe con $y = r$ impostando per Y una verifica corretta $y^2 = r^2$. Sapendo invece a priori che $b = 1$, F potrebbe tentare l'impersonazione generando un r a caso e trasmettendo a Y $c = r^2x^{-1}$ al posto di $c = r^2$ (x è pubblico). Se il bit di asseverazione è $b = 1$ l'impersonazione ha successo, in quanto F può rispondere con $y = r$ (invece di rs) impostando per Y una verifica corretta $r^2 = r^2/x^{-1}x$. Se però il bit di asseverazione è $b = 0$ il tentativo di impersonazione viene smascherato, in quanto F non è in grado di recuperare la radice di x.

In altre parole F potrebbe aver successo nell'impersonazione *solo* conoscendo a priori il bit di asseverazione, il che gli consentirebbe di trasmettere i valori

di c e y adeguati alla frode. F ha dunque una probabilità pari a 1/2 di evitare di essere smascherato. Solo l'utente X è dunque in grado di rispondere correttamente ad entrambe le richieste di asseverazione $b = 0$ o $b = 1$. Richiedendo che la verifica 2.(d) sia soddisfatta per tutte le t iterazioni si ottiene una probabilità d'impersonazione pari a 2^{-t}, che per t sufficientemente grande mette al riparo dalle frodi.

Le procedure di verifica a conoscenza nulla si possono svincolare dalla loro struttura interattiva, separando la nozione di sistema di verifica interattiva da quella di protocollo a conoscenza nulla, approdando ai cosiddetti *protocolli non interattivi*; per una trattazione esauriente del problema rimandiamo a [13].

10.3 Integrità dei dati e firme numeriche

Nell'odierna società dell'informazione si fa uso massiccio di documenti che non hanno più la consistenza fisico-cartacea di quelli tradizionali, ma che risiedono su sostrati facilmente alterabili da (neanche troppo) abili falsificatori. Questo vale tanto per i documenti conservati permanentemente negli archivi o basi di dati, quanto per quelli, ancora più esposti, in transito sui vari canali di comunicazione. Una volta stabilita la corretta identità dell'interlocutore è dunque necessario disporre di un meccanismo (o *firma numerica*) che svolga la funzione che un tempo spettava alla firma calligrafa, vergata in calce ai documenti cartacei, cioè quella di stabilire l'autenticità del documento e la sua non ripudiabilità da parte del firmatario.

Una differenza operativa rilevante tra firma calligrafa e firma numerica è che la prima non cambia quando impiegata su documenti diversi; le imperfezioni del meccanismo che porta all'esecuzione della firma da parte della stessa mano sono infatti tali da consentire contemporaneamente l'individuazione di caratteri ricorrenti (nel caso di una firma autentica) e la percezione dei caratteri incoerenti (per una firma falsa), rendendo plausibile l'uso iterato della stessa copia autentica di una firma.

Se per firma numerica intendiamo un'informazione in coda a un documento su supporto elettronico, non è in generale possibile costruire una firma numerica con le stesse caratteristiche di quella calligrafa. D'altra parte anche se lo fosse (esistono a tal riguardo delle penne "giroscopiche" che consentono di rilevare le velocità e le accelerazioni impresse sulla stessa penna e confrontarle con quelle del firmatario da verificare) ci sarebbe comunque il problema dell'alterazione del documento che la firma accompagna.

Si è pensato allora di costruire delle firme numeriche che siano legate al particolare documento in transito, inserendo delle informazioni di carattere contestuale e/o temporale. In tal modo l'acquisizione da parte di un falsificatore di una firma già impiegata precedentemente diventa del tutto inutile ai fini della falsificazione, poiché la verifica effettuata dal ricevente consente di leggere le informazioni contestuali inserite nella firma e di accorgersi del tentativo di frode.

10.3.1 Funzioni Hash

Per quanto detto precedentemente è opportuno realizzare un meccanismo di verifica dell'integrità dei dati contenuti in un documento, poiché anche una firma autentica potrebbe essere preceduta da un documento parzialmente alterato a insaputa del mittente (e del ricevente). Il problema è poi complicato dalla circostanza che molto spesso i documenti di mole rilevante devono essere frammentati in blocchi di dimensioni adeguate al sistema di cifratura impiegato (il *DES* richiede p.es. blocchi di 64 bit); poiché non è immaginabile che ciascun blocco sia firmato individualmente, bisogna prevedere dei meccanismi di firma e di verifica dell'integrità che consentano di lavorare globalmente sul documento. Fra l'altro, anche immaginando di controllare e asseverare i singoli blocchi, nessuno esclude che questi possano essere permutati o che qualcuno di questi possa essere cancellato, modificando così la struttura globale del documento.

Anche nel campo delle firme numeriche e della verifica dell'integrità dei dati esistono numerosissime procedure, alcune delle quali consentono di trattare i due problemi in modo congiunto. In questa sede riferiremo solo su una delle possibili soluzioni, che è quella di ricavare *un estratto* del documento globale e di applicare la firma numerica all'estratto in vece del documento originale; a tal scopo si può usare lo schema seguente:

messaggio	m	lunghezza arbitraria
	$\Downarrow$	
estratto	$e = h(m)$	lunghezza costante
	$\Downarrow$	
firma	$f = f(e)$	
	$\Downarrow$	
da spedire	(m, f)	

Poiché l'estratto ha di norma una dimensione molto contenuta (per qualsivoglia lunghezza del documento d'ingresso), l'effettuazione della firma risulta agevole. La verifica dell'integrità del documento si basa invece sul fatto che l'estratto viene ricavato mediante l'impiego di una funzione *Hash*. Quest'ultima accetta in ingresso una stringa m di dimensione arbitraria n e fornisce in uscita *l'estratto* (*Hash*) $e = h(m)$, di lunghezza costante k. Naturalmente se $k < n$ non si può avere corrispondenza biunivoca, e ci saranno in generale *più valori* dell'ingresso m che portano a *uno stesso* valore della funzione *Hash*. In tal caso si parla di *collisioni* per la funzione *Hash*. Tuttavia se la funzione *Hash* è ben progettata (cioè se la d.p. di uscita è sostanzialmente uniforme), la probabilità di queste collisioni risulta molto bassa, in quanto ci saranno mediamente 2^{n-k} ingressi distinti che portano alla stessa uscita; in altre parole, presi due ingressi a caso questi porteranno alla stessa uscita con una probabilità pari a (circa) 2^{-k}.

Ciò significa che la funzione *Hash* è in grado di costruire un *estratto rappresentativo del messaggio* m, ed è su questo estratto che va posta la firma numerica.

Un estratto consente inoltre di stabilire l'integrità del messaggio corrispondente usando il seguente procedimento:

Verifica d'integrità. Si supponga di avere m e il suo estratto *Hash* $e = h(m)$, che viene custodito in un posto sicuro nel quale non possa subire alterazioni. In qualunque momento successivo (o anche dall'altra parte del canale, purché si disponga di e) si può verificare se il messaggio m' è stato violato, calcolando $e' = h(m')$. Se $e = e'$ si accetta l'integrità di $m = m'$.

In generale le funzioni *Hash* impiegate in ambito crittografico per la verifica dell'integrità dei dati devono possedere il seguente requisito

Def. 10.1. *Una funzione Hash è a collisione intrattabile se è il problema di trovare due messaggi m e $m' \neq m$ tali che $h(m) = h(m')$ è computazionalmente intrattabile.*

È inoltre apprezzabile che la modificazione di anche uno solo dei bit del messaggio porti a un estratto "molto" diverso e che la funzione *Hash* si possa calcolare in modo efficiente.

Nella figura 8.2 viene riportata la struttura di base di una funzione *Hash* di tipo iterativo. In ingresso viene presentato l'intero messaggio m di lunghezza

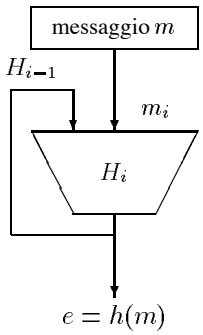


Figura 10.1 Struttura generale di una funzione *Hash* di tipo iterativo.

arbitraria, che viene segmentato in t blocchetti $m_1, m_2, \ldots m_t$ di lunghezza costante k. I blocchetti successivi costituiscono gli ingressi della i-esima iterazione della funzione H_i, la cui uscita dipende, oltre che da m_i, anche da H_{i-1}. L'iterazione consente di ottenere un estratto $e = h(m)$ che dipende da tutti i passi della medesima, e quindi dall'intero documento.

I cifrari a blocco studiati precedentemente consentono una facile attuazione della struttura testé descritta. Usando p.es. il *DES* si possono realizzare due tipologie di funzioni *Hash*, a seconda che si voglia integrare o meno l'autenticazione (iterazioni *Hash* con e senza chiave)

Iterazione *Hash* senza chiave

$$H_0 = I \qquad H_i = DES_{H_{i-1}}(m_i) \oplus m_i \qquad 1 \leq i \leq t$$

Iterazione *Hash* con chiave

$$H_0 = I \qquad H_i = DES_K(m_i \oplus H_{i-1}) \qquad 1 \leq i \leq t$$

dove $H_0 = I$ corrisponde a una inizializzazione e K alla chiave *DES*. Naturalmente in quest'ultimo caso la chiave K deve essere condivisa anche da chi effettua la verifica d'integrità a partire dall'estratto.

10.3.2 Firme numeriche

Una firma numerica su un messaggio è dunque un'operazione che si avvale della conoscenza di un'informazione privata, legata all'identità del firmatario, e di un'informazione associata al contesto nel quale la firma viene impressa, e quindi in sostanza legata al messaggio. La firma deve non deve essere ripudiabile dal firmatario, nel senso che il ricevente deve poter convincere un giudice imparziale sull'autenticità della stessa, senza peraltro che il giudice abbia accesso all'informazione privata.

Passeremo ora in breve rassegna alcuni metodi per effettuare firme numeriche, che si basano tutti su procedimenti a chiave pubblica già analizzati precedentemente. Di solito si effettua una distinzione tra le seguenti tipologie di firme numeriche:

Firma senza allegato. La verifica può essere fatta senza la conoscenza del messaggio; quest'ultimo si può ricavare direttamente dalla firma.

Firma con allegato. Richiedono la presenza del messaggio per effettuare la verifica.

In pratica la prima classe prevede che le firme riguardino messaggi piuttosto brevi e a lunghezza costante, mentre la seconda è associata a tecniche di tipo *Hash* su messaggi a lunghezza variabile.

Lo schema di firma *RSA* che tratteremo fra poco è un esempio di firma senza allegato, e consiste sostanzialmente nella semplice inversione dei ruoli di chiave di cifratura e decifrazione visti nello schema di cifratura *RSA* del paragrafo 9.3.2. Il mittente che deve apporre la firma applica al messaggio la propria chiave di *decifrazione* D_k, che è privata; chiunque può verificare la firma impiegando la chiave pubblica C_k.

Firma numerica RSA

1. Si generano due primi elevati p e q (usando le verifiche di primalità di *Solovay-Strassen* o *Miller-Rabin* viste nel paragrafo 9.2.4), producendo

$$n = pq$$

2. Si calcola $\Phi(n) = n\left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right) = (p-1)(q-1)$.

3. Si sceglie x coprimo con $\Phi(n)$ e se ne calcola il reciproco $\mod \Phi(n)$ mediante il teorema di Eulero (9.22); esistenza e unicità sono garantite dal fatto che $(x, \Phi(n)) = 1$

$$xy \equiv 1 \mod (\Phi(n)) \qquad (x, \Phi(n)) = 1$$

4. n e y costituiscono la chiave pubblica;
   $\Phi(n), p, q, x$ sono la chiave privata.

5. Se $m \in \boldsymbol{Z_n}$ è il messaggio, firma e verifica si eseguono nel modo seguente

$$\begin{array}{ll} \text{Firma} & f_x = m^x \mod n \\ \text{Verifica} & m = f_x^y \mod n \end{array} \tag{10.1}$$

Naturalmente, affinché il sistema funzioni correttamente è necessario che il messaggio contenga informazioni ridondanti, che possano consentire di escludere l'uso non autorizzato della stessa firma f_x in un momento successivo. Per i problemi legati alla sicurezza del sistema rimandiamo a quanto già detto per il sistema di cifratura *RSA* (vedi paragrafo 9.3.2).

Poiché la procedura vista non consente la segretezza, è possibile coniugare la firma numerica con una cifratura secondo lo schema già visto nella (7.5).

Anche il cifrario di *Rabin* si presta all'inversione nel ruolo delle chiavi, che porta all'uso della D_k come firma numerica associata al messaggio m. In questo caso bisogna però fare attenzione al fatto che lo spazio dei messaggi che possono essere firmati viene ristretto all'insieme Q_n dei resti quadratici $\mod n$.

Firma numerica di *Rabin*

1. Si generano due primi elevati p e q ($p, q \equiv 3 \mod 4$) usando le verifiche di primalità di *Solovay-Strassen* o *Miller-Rabin* viste nel paragrafo 9.2.4, producendo

$$n = pq$$

2. La *chiave pubblica* è data da n;
   la *chiave privata* è data dalla fattorizzazione p, q di n.

3. Se m è un messaggio ($m \in Q_n$), il mittente X deve calcolare la radice quadrata f_X di $m \mod n$, che costituisce la firma numerica.

4. La *verifica* richiede di effettuare l'operazione

$$m = f_X^2 \mod n$$

Si noti che nel punto 3, a seguito di quanto visto nel paragrafo 9.2.5, si ottiene la radice in tempo polinomiale mediante le (9.17), noti che siano p e q. Ciò porta all'individuazione di 4 soluzioni, e il mittente può selezionare liberamente una delle quattro soluzioni come firma effettiva.

Entrambi gli schemi visti precedentemente sono senza allegato. Lo schema seguente, che deriva direttamente dal cifrario di *ElGamal*, costituisce invece un esempio di firma con allegato; esso è particolarmente importante poiché costituisce la base per lo standard di firma numerica denominato *DSA* . Lo schema di firma numerica di *ElGamal* conserva la caratteristica di schema *stocastico*, propria anche del sistema di cifratura da cui deriva.

Fermi restando la costruzione di p e α, secondo quanto già descritto nel paragrafo 9.3.4, la firma e la sua verifica si effettuano secondo il seguente algoritmo.

Firma numerica di *ElGamal*

1. L'utente X sceglie un esponente $0 \leq x \leq p-2$ e calcola

$$\alpha^x \mod p$$

   che viene pubblicato.

2. La *chiave pubblica* di X è data da p, α, α^x;
   la *chiave privata* di X è data da x.

3. Se X vuole firmare il messaggio m ($0 \leq m \leq p-1$) deve selezionare in modo aleatorio un intero k coprimo con $p-1$, ($1 \leq k \leq p-2$). Poi calcola

   (a) $f_1 = \alpha^k \mod p$
   (b) $k^{-1} \mod (p-1)$
   (c) $f_2 = k^{-1}(m - xf_1) \mod (p-1)$
   (d) La firma di X è $f_X = (f_1, f_2)$

4. La verifica avviene controllando la validità della seguente uguaglianza

$$(\alpha^x)^{f_1} (f_1)^{f_2} = \alpha^m$$

La verifica consente di certificare la firma in quanto se f_X è stata prodotta effettivamente da X moltiplicando la 3.(c) ambo i lati per k si ottiene $xf_1 + kf_2 \equiv m \mod (p-1)$, e dunque

$$\alpha^m = \alpha^{xf_1+kf_2} \equiv (\alpha^x)^{f_1}\,(f_1)^{f_2} \mod (p-1)$$

Si noti che, nel caso in cui il messaggio sia molto grande, al posto di m si può usare un suo estratto $h(m)$ ricavato da una funzione *Hash*.

Esempio 10.1. Prendiamo i dati dell'esempio 9.4, cioè $p = 163$ e $\alpha = 7$ primitivo di $\boldsymbol{Z^*_{163}}$. L'utente X sceglie $x = 93$ come chiave privata, e pubblica $\alpha^x = 7^{93} \equiv 98 \mod 163$. Se X vuole firmare il messaggio $m = 26$, deve scegliere un valore aleatorio k coprimo con 162, p.es. $k = 53$, calcolare il reciproco $k^{-1} \equiv 107 \mod 162$ mediante il teorema di Eulero, e produrre

$$\left.\begin{aligned} f_1 &= 7^{53} \equiv 108 \mod 163 \\ f_2 &= 107\,(26 - 93\,108) \equiv 28 \mod 162 \end{aligned}\right\} \quad \Longrightarrow \quad f_X = (108, 28)$$

Ricevuto $f_X = (108, 28)$ si effettua la verifica impiegando il messaggio $m = 26$

$$\begin{aligned} 98^{108}\,108^{28} &\equiv 15 \mod 163 \\ 7^{26} &\equiv 15 \mod 163 \end{aligned}$$

Poiché sussiste l'uguaglianza la verifica ha esito positivo, la firma viene considerata vera e il messaggio viene attribuito a X.

10.3.3 Teoria dell'autenticazione

Nel paragrafo 7.1.2 si è ricordato che segretezza e autenticazione sono caratteri chiaramente distinguibili e indipendenti. Cercheremo ora di giustificare tale affermazione, mostrando inoltre che *una protezione assoluta nei confronti dell'impersonazione non è attuabile*. Nel modello che seguiremo, dovuto a Simmons [81], si suppone che l'utente X abbia concordato con Y una chiave segreta k e un sistema di cifratura C_k; X spedisce a Y il crittogramma $c = C_k(m)$. Il problema è quello di comprendere se l'avvenuta decifrazione secondo la chiave k costituisca garanzia assoluta sull'autenticità del messaggio (e quindi del mittente).

Un frodatore F può tentare d'ingannare Y con un *attacco d'impersonazione*, trasmettendo cioè a Y un crittogramma falso c' nella speranza che Y lo accetti. Un'altra possibilità è che F sostituisca un crittogramma legale c, in transito sul canale, con un proprio crittogramma falso c', attuando così un *attacco di sostituzione*.

In generale, dette rispettivamente P_I e P_S le probabilità d'impersonazione e di sostituzione, la strategia del frodatore sarà quella di scegliere l'attacco che

gli consente una probabilità più alta di successo; ha dunque senso definire la *probabilità di frode* come

$$P_F = \max(P_I, P_S) \tag{10.2}$$

Il nostro obiettivo è quello di trovare una limitazione inferiore per P_F.

Fissati gli insiemi $\mathcal{M}$, $\mathcal{K}$, $\mathcal{C}$, cioè gli spazi dei messaggi, delle chiavi e dei crittogrammi, una volta che X e Y abbiano deciso la chiave $k \in \mathcal{K}$ da usare la decifrazione va fatta usando sempre questa chiave. D'altra parte, per effetto della $c = C_k(m)$, per ogni k usata ci sono almeno $|\mathcal{M}|$ crittogrammi distinti c tali che $p(c/k) \neq 0$. Dunque, se il frodatore sceglie a caso un crittogramma c' da spedire a Y, la sua impersonazione ha una probabilità di successo limitata inferiormente dal rapporto

$$P_I \geq \frac{|\mathcal{M}|}{|\mathcal{C}|} \tag{10.3}$$

La limitazione, pur nella sua semplicità, evidenzia un fatto molto importante, e cioè che una protezione totale ($P_I = 0$) nei confronti dei giochi d'impersonazione *non è attuabile*. L'unica strategia possibile è quella di aumentare $|\mathcal{C}|$.

La limitazione (10.3) può essere però migliorata; nel seguito faremo uso della dimostrazione riportata in [45] e dovuta a Körner.

Per isolare le condizioni nelle quali si verifica un successo nell'impersonazione, definiamo la *funzione di autenticazione*

$$\chi(c,k) = \begin{cases} 1 & \text{se } \exists\, m : C_k(m) = c \\ 0 & \text{se } \nexists\, m : C_k(m) = c \end{cases}$$

che vale 1 quando il crittogramma c viene autenticato mediante la chiave k, 0 altrimenti. La strategia del falsificatore sarà quella di scegliere il crittogramma c che rende massima la probabilità di una decifrazione significativa, cioè la

$$P(c \text{ valido}) = \sum_k \chi(c,k)p(k) \tag{10.4}$$

e dunque

$$P_I = \max_c \{P(c \text{ valido})\} = \max_c \sum_k \chi(c,k)p(k) \tag{10.5}$$

In tal caso vale il seguente

Lemma 10.1 (Limitazione di Simmons). *La probabilità d'impersonazione è limitata inferiormente da*

$$P_I \geq 2^{-I(C \wedge K)} \tag{10.6}$$

con $I(C \wedge K)$ *mutua informazione tra le v.a.* C *e* K.

Dim. Dall'espressione (2.5) della mutua informazione possiamo scrivere, tenendo conto che $p(k/c) = 0$ se $\chi(c,k) = 0$

$$I(C \wedge K) = \sum_c p(c) \sum_k p(k/c) \log \frac{p(k/c)}{p(k)} =$$
$$= \sum_c p(c) \sum_k \chi(c,k) p(k/c) \log \frac{\chi(c,k) p(k/c)}{\chi(c,k) p(k)}$$

Definiamo ora $a_k = \chi(c,k)p(k/c)$, $b_k = \chi(c,k)p(k)$. Ricordando poi che $\sum_k a_k = 1$ e che dalla (10.4) si ricava $\sum_k b_k = P(c \text{ valido})$, si può applicare la disuguaglianza della somma logaritmica (2.2), ottenendo

$$I(C \wedge K) \geq -\sum_c p(c) \log P(c \text{ valido}) \geq$$
$$\geq -\max_c \log P(c \text{ valido}) = -\log P_I$$

che è la tesi □

Teorema 10.1 (Limitazione di Johannesson-Sgarro). *La probabilità d'impersonazione è limitata inferiormente da*

$$P_I \geq 2^{-\inf I(C \wedge K)} \tag{10.7}$$

dove $\inf I(C \wedge K)$ *è calcolato su tutte le statistiche della sorgente dei messaggi che non alterano* $\chi(c,k)$.

Dim. Anche se le $P(c \text{ valido})$ e P_I sono indipendenti dalla statistica della sorgente, ciò non succede per la $I(C \wedge K)$, che si può dunque minimare variando la d.p. dei messaggi. □

Oss. 10.1. La limitazione (10.3) si può ricavare direttamente dalla (10.7) nel caso di cifratura deterministica. Infatti

$$I(C \wedge K) = H(C) - H(C/K) = H(C) - H(M) \leq \log |\mathcal{C}| - H(M)$$

e dunque

$$\inf I(C \wedge K) \leq \log |\mathcal{C}| - \log |\mathcal{M}|$$

cioè la (10.3) non appena si sostituisca il tutto nella (10.7).

Bibliografia

[1] Aczél J. e Daróczy Z., *On Measures of Information and their Characterizations*, Mathematics in Science and Engineering, vol. 115. Academic Press, New York-London, 1975.

[2] Aho A., Hopcroft J. e Ullman J., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass. 1974.

[3] Ash R. B., *Information Theory*, Dover Publications, Inc., New York, 1990.

[4] Barg A., Guritman S. e Simonis J., "Strengthening the Varšamov-Gilbert Bound", *DIMACS Technical Report* 98-41, 1998.

[5] Beker H. e Piper F., *Cipher Systems-The protection of Communications*, Northwood Book, London, 1982.

[6] Berardi L., *Algebra e teoria dei codici correttori*, Franco Angeli, Milano, 1994.

[7] Berardi L. e Beutelspacher A. *Crittologia*, Franco Angeli, Milano, 1996.

[8] Berger T., *Rate Distortion Theory*, Prentice Hall, Englewood Cliffs, 1971.

[9] Berlekamp E., *Algebraic Coding Theory*, Mc Graw Hill, New York, 1968.

[10] Berson T.A. "Failure of the McEliece Public-Key Cryptosystem Under Message-Resend and related-Message Attack", *CRYPTO '97*, LNCS 1294, Springer Verlag, 1998.

[11] Beth T. e Duo Dai Z., "On the Complexity of Pseudo-Random Sequences - or: If You Can Describe A Sequence It Can't Be Random", *Eurocrypt '89*, LNCS 434, Springer Verlag, pp. 533-543, 1990.

[12] Boneh D. e Venkatesan R., "Breaking RSA May Not Be Equivalent to Factoring", *Eurocrypt '98*, LNCS 1403, Springer Verlag, 1998.

[13] Blum M. e De Santis A., S. Micali, G. Persiano, "Noninteractive Zero-Knowledge", *SIAM Journal on Computing*, n.20, pp. 1084-1118, 1991.

[14] Borst J., Knudsen L.R. e Rijmen V., "Two Attack on Reduced IDEA", *Eurocrypt '97*, LNCS 1233, Springer Verlag, 1998.

[15] Capocelli R.M. e De Santis A., "New Bounds on the Redundancy of Huffman Codes", *IEEE Transactions on Information Theory*, 37 (1991), no. 4, pp.1095-1104.

[16] Chaitin G.J., "On the Length of Programs for Computing Binary Sequences", *J. Assoc. Comp. Mach.*, vol. 13, pp. 547-569, 1966.

[17] Chor B. e Rivest R.L., "A Knapsack-Type Public Key Cryptosystem Based on Arithmetic in Finite Fields", *CRYPTO '84*, LNCS 196, Springer Verlag, pp. 54-65, 1985.

[18] Cohn, P.M., *Algebra*, Wiley, New York, 1982.

[19] Cover T.M. e Thomas J.A., *Elements of Information Theory*, John Wiley & Sons, New York, 1991.

[20] Csiszár I., "The method of Types", Commemorative Issue, 1948-1998, *IEEE Transactions on Information Theory*, vol. IT-44, n.6, pp.2505-2523, 1998.

[21] Csiszár e I. and Körner J., *Information Theory: Coding Theorems for Discrete Memoryless Systems*, Academic Press, New York, 1981.

[22] Daboni L., *Calcolo delle probabilità ed elementi di statistica,* UTET, Torino, 1992.

[23] Denning D.E., *Cryptography and Data Security*, Addison Wesley, Massachusetts, 1983.

[24] De Prisco R. e De Santis A,. "A New Bound for the Data Expansion of Huffman Codes", *IEEE Transactions on Information Theory*, vol. IT-43 , n. 6, 2028–2032, 1997.

[25] Diffie W. e Hellman M.E., "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. IT-22, pp.644-654, 1976.

[26] Elia M., "Some Results on the Existence of Binary Linear Codes", *IEEE Transactions on Information Theory*, IT-29, No.6, Nov. 1983.

[27] Fabris F., "Variable-Length to Variable-Length Source Coding: A Greedy Step-by-Step Algorithm", *IEEE Transactions on Information Theory*, vol. IT-38, n.5, pp. 1609-1617, 1992.

[28] Fabris F., "Sharpening the Gilbert-Varšamov Bound in the Finite Case", *Journal of Discrete Mathematical Sciences & Cryptography*, vol. n.4, n.1 January 2001, pp.1-11.

[29] Fabris F., "Stream Ciphering Techniques Based on n-Tuples Juxtaposition Sequences", *Journal of Information & Optimization Sciences*, vol.10, n.2, May 1989, pp.309-335.

[30] Fabris F., "A Lower Bound on the Expected Complexity in the Set of n-Tuples of Given Weight", *Journal of Information & Optimization Sciences*, vol.11, n.2, May 1990, pp.249-262.

[31] Fabris F., Sgarro A. e Pauletti R., "Tunstall Adaptive Coding and Miscoding", *IEEE Transactions on Information Theory*, vol. IT-42, n.6, pp. 2167-2180, 1996.

[32] Fabris F. e Sgarro A., "On the Composition of Tunstall Messages", *IEEE Transactions on Information Theory*, vol. IT-45, n.5, pp.1608-1612 , 1999.

[33] Facchini A., *Algebra e Matematica Discreta*, Zanichelli, Bologna, 2000.

[34] Feller W., *An Introduction to Probability and its Applications*, Wiley, New York, 1968.

[35] Fredricksen H., "A Survey of Full Length Non Linear Shift Register Cycle Algorithms", *SIAM Review*, vol.24, pp.195-221, 1982

[36] Gallager R.G., *Information Theory and Reliable Communication*, J. Wiley, New York, 1968.

[37] Garey M.R. e Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[38] Gargano L., Körner J. e Vaccaro U. "Capacities: from Information Theory to Extremal Set Theory", *J. Combin. Theory Ser. A*, 68, no. 2, pp. 296-316, 1994.

[39] Geffe P. "How to Protect Data With Ciphers that are really Hard to Break", *Electronics*, 46, pp.99-101, 1973.

[40] Golomb S.W., *Shift Register Sequences*, Holden Day, San Francisco, 1967.

[41] Gray R. M., *Entropy and Information Theory*, Springer-Verlag, New York, 1990. xxiv+332 pp.

[42] Hankerson D., Harris G.A. e Johnson P.D., *Introduction to Information Theory and Data Compression*, CRC Press, Boca Raton, FL, 1998.

[43] Hashim A.A., "Improvement on Varšamov-Gilbert lower bound on Minimum Hamming Distance of Linear Codes", *Proc. Inst. Elec. Engrs.*, 125, 1978, no. 2, 104-106.

[44] *IEEE Transactions on Information Theory*, Commemorative Issue, 1948-1998, vol. IT-44, n. 6, 1998.

[45] Johannesson R. e Sgarro A., "Strengthening Simmons' bound on Impersonation"*IEEE Transactions on Information Theory*, vol. IT-37, pp.1182-1185, 1991.

[46] Kahn D. *The Codebreakers*, MacMillan, New York, 1967.

[47] Kampé de Fériet J. "Théorie de l'Information. Principe du Maximum de l'Entropie et ses Applications à la Statistique et à la Mecanique", *Laboratoire de Calcul de la Faculté des Sciences de l'université de Lille*, Lille, 1963.

[48] Kampé de Fériet J. e Forte B., "Information et Probabilité ", I,II,III. *C.R. Acad. Sci. Paris*, serie A 265, 110-114, 142-146, pp. 350-353, 1967.

[49] Knuth D.E., *The Art of Computer programming*, vol.I-II-III, Addison Wesley, Massachusetts, 1973.

[50] Koblitz N., *A Course in Number Theory and Cryptography*, Springer-Verlag, New-York, 1994.

[51] Kolmogorov A.N., "Three Approaches to the Quantitative definition of information", *Problems of Information Transmission*, vol.1, pp. 4-7, 1965.

[52] Kolmogorov A.N., "Logical Basis for Information Theory and Probability Theory", *IEEE Transactions on Information Theory*, vol. IT-14, pp. 662-664, 1968.

[53] Kullback S., *Information Theory and Statistics*, Dover Publications, Mineola, NY, 1997.

[54] Lelewer D.A. e Hirschberg D.S., "Data Compression", *ACM Computing Surveys*, vol.19, n.3, 1987.

[55] Longo G.O., *Teoria dell'informazione*, Boringhieri, Torino, 1980.

[56] Longo G.O., *Il nuovo Golem*, Laterza, Bari, 1998.

[57] Longo G.O., *Homo technologicus*, Meltemi , Roma, 2001.

[58] Longo G.O. e Galasso G., "An Application of the Informational Divergence to Huffman Codes", *IEEE Transactions on Information Theory*, vol. IT-28, n.1, pp. 36-43, 1982.

[59] MacWilliams F.J. e Sloane N.J.A., *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[60] Massey J.L., "Shift Register Sequences and BCH Decoding", *IEEE Transactions on Information Theory*, vol. IT-15, pp.122-127, 1969.

[61] Matsui M., "Linear Cryptanalysis Method for DES Cipher", *Eurocrypt '93*, LNCS 765, Springer Verlag, pp. 386-397, 1994.

[62] Maurer U., "A Universal Statistical Test for Random Bit Generators", *CRYPTO '90*, LNCS 537, Springer-Verlag, pp.409-420, 1991.

[63] McEliece R.J., Rodemich E.R., Rumsey H.C. e Welch L.R., "New Upper Bounds on the Rate of a Code via the Delsarte-MacWilliams Inequalities", *IEEE Transactions on Information Theory*, IT-23, No.2, pp. 157-165, Mar. 1977.

[64] Menezes A. J., van Oorschot P.C. e Vanstone S. A., *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1997.

[65] Nemetz T. e Simon J., "Self Information and Optimal Codes", in *Proc. Colloq. Infor. Theory.*, Keszthely, Hungary, pp. 457-468, Aug. 1975.

[66] Peterson W. W e Weldon E. J., *Error-Correcting Codes*, M.I.T. Press, Cambridge, Mass.-London, 1972.

[67] Piret P., *Convolutional Codes - an Algebraic Approach*. MIT Press, Cambridge, MA-London, 1988.

[68] Pless, V.S e Huffman W.C., (a cura di) *Handbook of Coding Theory*, vol. I e vol II, Elsevier, Ansterdam, 1998.

[69] Pretzel O., *Error-Correcting Codes and Finite Fields*, Clarendon Press, Oxford, 1992.

[70] Rogaway P. e Coppersmith D., "A Software-Optimized Encryption Algorithm", *Fast Software Encryption*, Cambridge Security Workshop, LNCS, Springer-Verlag, 56-63, 1994.

[71] Rüeppel R.A., *Analysis and Design of Stream Cipher*, Springer Verlag, New York, 1986.

[72] Salomaa A. *Public-Key Cryptography*, EATCS, Springer Verlag, Berlin, 1990.

[73] Serafini P., *Ottimizzazione*, Zanichelli, Bologna, 2000.

[74] Sgarro A., "An Informational Divergence Geometry for Stochastic Matrices", Calcolo, vol.15, pp.41-49, 1978.

[75] Sgarro A., *Crittografia. Tecniche di protezione dei dati riservati*, Muzzio, Padova, 1986.

[76] Shamir A., "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem", *CRYPTO '82*, pp. 279-288, 1983.

[77] Shannon C.E., "A mathematical Theory of Communication", *Bell System Technical Journal*, 27, pp. 379-423 (pt.1), pp.623-656 (pt.2), 1948.

[78] Shannon C.E., "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, 28, pp.656-715 (pt.2), 1949.

[79] Siegenthaler T., "Correlation-Immunity of Non-Linear Combining Functions for Cryptographic Applications", *IEEE Transactions on Information Theory*, vol. IT-30, pp.776-780, 1984.

[80] Simmons G.J., (a cura di) *Contemporary Cryptology - The Science of Information Integrity*, IEEE Press, New York, 1992.

[81] Simmons G.J.,"Authentication Theory/Coding Theory", *CRYPTO '84*, LNCS 196, Springer Verlag, pp. 411-431, 1985.

[82] Stinson D. R., *Cryptography - Theory and Practice*, CRC Press, Boca Raton, FL, 1995.

[83] Solomonoff R.J., "A Formal Theory for Inductive Inference", *Information & Control*, vol. 7, pp. 1-22, 224-254, 1964.

[84] van Lint J.H., *Introduction to Coding Theory*, GTM, Springer Verlag, 1982.

[85] Viterbi A. e Omura J., *Digital Communication and Coding*, McGraw-Hill, New Tork, 1978.

[86] Welsh D., *Codes and Cryptography*, Clarendon Press, Oxford, 1988.

[87] Ziv J. e Lempel A., "A Universal Algorithm for sequential Data Compression", *IEEE Transactions on Information Theory*, vol. IT-23, pp.337-343, 1977.

[88] Ziv J. e Lempel A., "Compression of Individual Sequences by Variable Rate Coding", *IEEE Transactions on Information Theory*, vol. IT-24, pp.530-536, 1978.

Indice analitico

Francesco Fabris

Teoria dell'informazione, codici, cifrari

Questo testo è stato pensato come sussidio didattico per gli studenti delle lauree di I e II livello relative alle classi di Scienze e Tecnologie informatiche e di Ingegneria dell'Informazione. È suddiviso in tre parti, dedicate rispettivamente agli aspetti normativi della teoria dell'informazione, allo studio della teoria dei codici di sorgente e dei codici correttori d'errore, e alla comprensione dei modelli matematici che sono alla base delle moderne tecniche crittografiche per la protezione (attiva e passiva) dei dati mediante cifrari. La trattazione congiunta, in un unico libro, di tali argomenti si giustifica in base alla considerazione che tanto la teoria dell'informazione (e dei codici) quanto la crittografia devono il loro assetto normativo attuale ai lavori di Claude E. Shannon, che dotò di un modello matematico efficace e stabile sia il processo di trasmissione dell'informazione sia quello relativo alla protezione della stessa nei confronti di utenti non autorizzati. Dal punto di vista didattico, la struttura modulare del volume (corrispondente a 3 moduli da 4 crediti o a 2 moduli da 6 crediti) lo rende adatto all'insegnamento sia nel triennio della laurea breve sia nel biennio della laurea specialistica.

Francesco Fabris insegna Teoria dell'Informazione presso la Facoltà di Scienze dell'Università di Udine. Autore di numerose pubblicazioni a livello internazionale, collabora tra l'altro con «IEEE Transactions on Information Theory», la rivista dell'Institute of Electrical and Electronic Engineers, di cui è membro.

ISBN 88-339-5661-X

9 788833 956619

LIRE 50.000 (I.I.)

€ 25,82