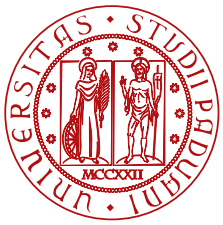

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Cenni di teoria dell'Informazione di Shannon

Relatore
Prof. Mariconda Carlo

Laureando
Malvestio Andrea

ANNO ACCADEMICO 2024-2025

Data di laurea 10/03/2025

Sommario

La teoria dell'informazione rappresenta una delle pietre miliari dell'informatica e delle telecomunicazioni. Essa fornisce strumenti matematici indispensabili per poter analizzare la trasmissione dei dati, valutare l'efficienza dei sistemi di codifica e per poter quantificare l'informazione. Questa tesi ha lo scopo di studiare alcune delle nozioni principali della teoria dell'informazione collegando i fondamenti teorici alle applicazioni pratiche nella crittografia.
Dopo una panoramica storica che parte dalle prime tecniche di cifratura e l'evoluzione delle metodologie di crittanalisi, si studiano alcune tecniche che hanno segnato il passaggio da sistemi semplici a sistemi moderni, evidenziando l'importanza degli aspetti statistici e matematici.
L'obiettivo principale di questo elaborato è quello di evidenziare il forte legame che c'è tra la teoria dell'informazione e la crittografia, illustrando come i modelli matematici possano contribuire a una migliore comprensione e a un'innovazione continua nei sistemi di sicurezza delle comunicazioni.

Indice

Introduzione

La necessità di proteggere le informazioni trasmesse risale all'antichità. Uno dei primi esempi noti di crittografia è il cifrario di Cesare che consiste in un sistema di sostituzione piuttosto semplice nel quale ogni lettera del messaggio originale veniva spostata di un numero fisso di posizioni all'interno dell'alfabeto. Questo metodo, nonostante fosse rudimentale, metteva già in evidenza l'importanza della segretezza nelle comunicazioni.
Con il passare del tempo, le tecniche di cifratura sono diventate sempre più sofisticate. Nel Rinascimento, con il cifrario di Vigenère, venne infatti introdotto il concetto di chiave variabile per ostacolare la decrittazione non autorizzata. A lungo fu considerato indecifrabile, ma successivamente riuscirono a violarlo analizzando la frequenza di comparsa delle lettere. Fu uno dei primi metodi a dimostrazione di come le regolarità statistiche potessero essere sfruttate per decriptare i testi cifrati.
Nel XX secolo con l'avvento di macchine crittografiche come l'Enigma, utilizzata dai tedeschi durante la Seconda Guerra Mondiale, la crittografia divenne parte integrante nelle operazioni militari. La decifrazione di Enigma, da parte di Alan Turing e del team Bletchley Park, rappresentò non solo una svolta nel corso della guerra ma anche l'inizio della crittoanalisi moderna attraverso l'impiego di modelli matematici per la decrittazione.
Parallelamente, lo sviluppo della teoria matematica dell'informazione ha fornito gli strumenti per comprendere e ottimizzare i sistemi di comunicazione. Nel 1948 con la pubblicazione del suo articolo *"A Mathematical Theory of Communication"*[3], Shannon[1] riuscì a dimostrare che l'informazione poteva essere quantificata in termini di probabilità e che la trasmissione dei dati era soggetta a limiti imposti dal rumore e dalla capacità del canale. Per raggiungere tale scoperta ha introdotto concetti fondamentali come l'entropia, la ridondanza e la capacità di un canale di comunicazione. La teoria dell'informazione formulata da Shannon ha poi spianato la strada alla moderna elaborazione dei segnali, alla compressione dei dati e alle trasmissioni digitali, che oggi costituiscono la base delle telecomunicazioni globali.
Questo elaborato non si limita a descrivere in maniera discorsiva i concetti della crittografia e della teoria dell'informazione, ma si avvale degli strumenti matematici e di probabilità acqui-

[1] Claude Elwood Shannon (1916-2001) è stato un matematico e ingegnere elettrico americano che ha gettato le basi teoriche per i circuiti digitali e la teoria dell'informazione, un modello matematico della comunicazione.

siti nel corso di Fondamenti di Analisi Matematica e Probabilità per formalizzare e dimostrare rigorosamente i risultati.
Il presente elaborato è strutturato in tre capitoli.

- Capitolo 1: si analizza il cifrario di Vigenère, un classico esempio di cifrario polialfabetico risalente al sedicesimo secolo. Si procederà ad esaminare in dettaglio il suo funzionamento, evidenziando il ruolo fondamentale delle permutazioni e delle modalità operative impiegate nella cifratura. Verranno illustrati due metodi per individuare con rigore matematico la lunghezza della chiave: il Metodo Kasiski e l'uso dell'indice di coincidenza. Successivamente sarà presentato un procedimento per la determinazione della chiave nota la sua lunghezza basato sull'indice di coincidenza mutuo. In questo capitolo sono presenti le seguenti dimostrazioni: correttezza della formula dell'indice di coincidenza, valore dell'indice di coincidenza per pattern random e valore minimo, correttezza della formula dell'indice di coincidenza mutuo, valore massimo dell'indice di coincidenza mutuo.

- Capitolo 2: si studiano algoritmi di collisione e attacchi Meet-in-the-Middle, argomenti di grande rilevanza nel panorama della crittografia moderna. Essi sfruttano alcune vulnerabilità presenti nei problemi matematici per compromettere la sicurezza di sistemi crittografici. In particolare saranno presi in esame un algoritmo di collisione per il logaritmo discreto, e un teorema che fornisce un limite inferiore alla probabilità che si verifichi una collisione tra due liste di oggetti e la formula con cui è possibile calcolare tale probabilità. Sono fornite sia la dimostrazione del teorema, sia quella dell'algoritmo di collisione per il logaritmo discreto.

- Capitolo 3: dedicato alla teoria dell'informazione di Shannon, con particolare attenzione ai concetti che costituiscono le fondamenta della comunicazione digitale e della sicurezza dei dati, i quali vengono affrontati con un approccio matematico rigoroso. L'idea base consiste nel considerare in un sistema crittografico le chiavi, i messaggi e le loro cifrature come delle variabili aleatorie discrete. Inizialmente vengono fornite delle nozioni sui cifrari simmetrici per poi passare ad analizzare le varie proprietà dei sistemi che godono di segretezza perfetta, ognuna con annessa dimostrazione. Si affronta poi il concetto di entropia nell'ambito della teoria dell'informazione fornendone prima la definizione e poi le proprietà con le relative dimostrazioni. Successivamente si dimostra che ogni funzione di entropia che soddisfa le proprietà descritte in precedenza, rispetta una particolare forma. Per concludere la parte riguardante l'entropia si verificano i valori di minimo e massimo

di questa funzione. L'ultimo argomento trattato nel capitolo è l'applicazione dei concetti di ridondanza ed entropia al linguaggio naturale fornendo anche un limite superiore ed inferiore alla ridondanza ed il modo per interpretarne i valori. Qui l'idea di Shannon consiste nell'interpretare un linguaggio come un processo stocastico: un sistema che produce sequenze di simboli governato dalle leggi della probabilità.

Capitolo 1

Il cifrario di Vigenère

Il cifrario di Vigenère, inventato nel sedicesimo secolo, deve il suo nome a Blaise de Vigenère[1] che con il suo libro *"Traicté des chiffres, ou secretes manieres d'escrire"* introdusse un ingegnoso sistema chiamato *autokey system* nel quale la parola chiave scelta per cifrare il testo viene concatenata con il testo in chiaro formando così una chiave variabile.
Il cifrario di Vigenère appartiene alla categoria dei cifrari polialfabetici che, a differenza di quelli monoalfabetici (come ad esempio il cifrario di Cesare), utilizza più alfabeti di sostituzione per codificare il messaggio.

1.1 Funzionamento dei cifrari

Per analizzare il funzionamento generale dei cifrari è necessario prima individuare due componenti principali: permutazione e modalità di operazione.
Una permutazione è una funzione che trasforma un elemento (in questo caso parliamo di lettere) in modo tale che ognuno di questi abbia un inverso univoco. Una modalità di operazione invece è un algoritmo che utilizza una permutazione per elaborare messaggi di lunghezza arbitraria.

1.1.1 Permutazione

La maggioranza dei cifrari comuni si basa sulla sostituzione di ogni lettera con un'altra. Tuttavia, in un cifrario tale sostituzione non può essere arbitraria ma deve essere una riorganizzazione di tutte le lettere da A a Z, in modo tale da permettere ad ogni lettera di avere un inverso unico, ovvero una permutazione. In ogni caso ciò non è sufficiente; una delle caratteristiche più importanti per un cifrario è la sicurezza, una permutazione, per essere sicura, deve soddisfare questi tre criteri:

[1] Blaise de Vigenère (1523-1596) era un crittografo e diplomatico Francese.

1. La permutazione deve essere determinata dalla chiave, in modo da rendere difficile l'identificazione della permutazione utilizzata fintanto che la chiave resta segreta.

2. Chiavi diverse devono produrre permutazioni diverse. In caso contrario, se chiavi diverse producessero la stessa permutazione, esiterebbero meno chiavi distinte rispetto alle permutazioni distinte ed aumenterebbero quindi le probabilità di decifrare il testo in assenza della chiave.

3. La permutazione deve sembrare casuale. Deve essere difficile poter individuare schemi evidenti nel testo cifrato perché essi potrebbero rendere la permutazione prevedibile e quindi meno sicura.

1.1.2 Modalità di operazione

La modalità di operazione di un cifrario serve a ridurre il rischio di identificazione delle lettere che appaiono più volte nel testo in chiaro utilizzando permutazioni diverse per le diverse occorrenze della lettera. Supponiamo di avere una permutazione sicura che mappa la lettera S in K, A in J e O in L. Per esempio la parola "SASSO", se non venisse applicata una modalità di operazione, verrebbe cifrata come "KJKKL" nella quale si nota subito che la stessa lettera viene ripetuta tre volte. Se a questo si aggiungono altre informazioni sul messaggio cifrato come la categoria di appartenenza e si considerano solo le parole di cinque lettere, si riducono drasticamente le possibili parole associabili a quella determinata sequenza di caratteri cifrati.

1.2 Funzionamento del cifrario di Vigenère

Il cifrario di Vigenère funziona utilizzando diversi cifrari a sostituzione per cifrare lettere differenti. Per capire di quanto debba essere spostata ogni lettera, i due interlocutori, che per convenzione chiameremo Bob ed Alice, dovranno prima decidere una frase o una parola chiave. Successivamente Bob utilizzerà in successione le singole lettere della chiave per cifrare il messaggio che dovrà trasmettere ad Alice. Il modo in cui vengono sostituite le lettere è molto semplice, verrà considerata non tanto la lettera della parola chiave ma la relativa posizione all'interno dell'alfabeto. Per esempio, se la chiave scelta è la parola "casa", la prima lettera subirà uno spostamento di due posizioni, la seconda e la quarta rimarranno invariate, mentre la terza verrà spostata di diciotto posizioni.
Spesso però capita che la chiave individuata da Bob ed Alice sia più corta dell'intero messaggio da cifrare quindi è necessario ripeterla più volte fino a raggiungere la lunghezza dell'intero testo in chiaro.

Volendo per esempio cifrare la parola ”informatica” utilizzando sempre la chiave ”casa” conviene dividere in blocchi di quattro lettere (lunghezza della chiave) il testo in chiaro:

Testo	I	N	F	O	R	M	A	T	I	C	A
Chiave	C	A	S	A	C	A	S	A	C	A	S

Tabella 1.1: Testo originale e chiave ripetuta ciclicamente.

Successivamente è necessario convertire le lettere nel numero corrispondente alla loro posizione nell'alfabeto:

Lettera	I	N	F	O	R	M	A	T	I	C	A
Valore	8	13	5	14	17	12	0	19	8	2	0

Tabella 1.2: Conversione del testo in chiaro in valori numerici.

Lettera	C	A	S	A	C	A	S	A	C	A	S
Valore	2	0	18	0	2	0	18	0	2	0	18

Tabella 1.3: Conversione della chiave in valori numerici.

A questo punto sommiamo i valori ottenuti in modulo 26:

$$(8+2) \mod 26 = 10, \quad (13+0) \mod 26 = 13, \quad (5+18) \mod 26 = 23$$

$$(14+0) \mod 26 = 14, \quad (17+2) \mod 26 = 19, \quad (12+0) \mod 26 = 12$$

$$(0+18) \mod 26 = 18, \quad (19+0) \mod 26 = 19, \quad (8+2) \mod 26 = 10$$

$$(2+0) \mod 26 = 2, \quad (0+18) \mod 26 = 18$$

Ed infine convertiamo i numeri in lettere:

Valore	10	13	23	14	19	12	18	19	10	2	18
Lettera	K	N	X	O	T	M	S	T	K	C	S

Tabella 1.4: Conversione dei valori cifrati nelle lettere corrispondenti.

La parola ”informatica” cifrata con la chiave ”casa” diventa quindi ”KNXOTMSTKCS”. L’aspetto chiave di questo cifrario è che più occorrenze di una stessa lettera del testo in chiaro possono essere cifrate in più modi diversi. Tuttavia, come si può notare nell’esempio precedente, se la parola chiave è piuttosto breve, parti ripetute del testo in chiaro possono allinearsi con la stessa parte di chiave causando così delle ripetizioni nel testo cifrato (come è avvenuto per le lettere ”I” e ”A” che sono state convertite entrambe le volte rispettivamente in ”K” e ”S”). Questo problema può permettere ad un attaccante di capire la lunghezza della parola chiave e quindi di poter individuare altri eventuali schemi.

1.2.1 La Tavola di Vigenère

Esiste un altro metodo per cifrare e decifrare dei messaggi con il cifrario di Vigenère senza passare però per la conversione delle lettere in numeri. Questo metodo consiste nell’utilizzo della Tavola di Vigenère, anche detta Tabula Recta. Si tratta di una matrice quadrata di ordine 26 nella quale ad ogni riga corrisponde un alfabeto che differisce da quello immediatamente sopra per una traslazione di una signola posizione.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tabella 1.5: Tavola di Vigenère

La cifratura di un messaggio utilizzando questa tavola avviene seguendo questi semplici passi:

- trovare la lettera del testo in chiaro nella prima riga della tabella
- trovare la lettera della chiave nella prima colonna
- individuare la lettera cifrata che si troverà all'intersezione tra la colonna e la riga delle lettere trovate ai punti precedenti
- ripetere per ogni lettera del testo in chiaro.

Per decifrare il messaggio il procedimento è molto simile:

- trovare la lettera della chiave nella prima colonna
- trovare la lettera del messaggio cifrato nella riga corrispondente alla lettera trovata al punto precedente
- in cima alla colonna appena individuata ci sarà la lettera del testo in chiaro corrispondente.
- ripetere per ogni lettera del testo cifrato.

1.3 Crittoanalisi teorica del cifrario di Vigenère

Per molto tempo il cifrario di Vigenère è stato definito come "inviolabile", tuttavia, attraverso una serie di studi di crittoanalisi, qualche secolo più tardi si scoprì che ciò non era affatto vero. Per riuscire a fare breccia nel cifrario, il primo obiettivo da raggiungere è la determinazione della lunghezza della parola chiave. Questo obiettivo è stato conseguito da Friedrich Kasiski[2] con quello che verrà chiamato Metodo Kasiski, descritto nel suo libro *"Die Geheimschriften und die Dechiffrir-kunst"*[2]. Il suo metodo prevede l'identificazione di frammenti ripetuti nel testo cifrato e la compilazione di una lista tenendo conto della distanza fra le varie ripetizioni. Secondo Kasiski la lunghezza della chiave è con molta probabilità un divisore comune della maggior parte delle distanze individuate. Sarà necessario scartare le ripetizioni dovute al caso ma queste, essendo appunto casuali, non saranno divisibili per la lunghezza della chiave.
Un altro metodo per trovare la lunghezza della chiave è quello che sfrutta la tabella delle frequenze delle lettere in una determinata lingua (prenderemo in esame la lingua inglese), attraverso la quale è possibile notare immediatamente che alcune lettere sono molto più comuni di altre.
Un ulteriore strumento utilizzato in questa fase è l'indice di coincidenza. Si tratta di una misura statistica che, se calcolata su segmenti derivati da una suddivisione del testo in blocchi di

[2]Friedrich Kasiski (1805-1881) è stato un crittografo e un maggiore dell'esercito tedesco

dimensione pari alla lunghezza ipotizzata della chiave, permette di verificare se tale lunghezza è corretta.
Una volta trovata la lunghezza corretta, si passa alla fase di determinazione vera e propria della chiave. In questo processo viene impiegato l'indice di coincidenza mutuo, il cui calcolo permette di determinare le lettere della chiave analizzando le traslazioni tra i testi di un numero di blocchi pari alla lunghezza della chiave.

Lettera	Frequenza (%)	Lettera	Frequenza (%)
A	8.17	N	6.75
B	1.49	O	7.51
C	2.78	P	1.93
D	4.25	Q	0.10
E	12.70	R	5.99
F	2.23	S	6.33
G	2.02	T	9.06
H	6.09	U	2.76
I	6.97	V	0.98
J	0.15	W	2.36
K	0.77	X	0.15
L	4.03	Y	1.97
M	2.41	Z	0.07

Tabella 1.6: Frequenza delle lettere nella lingua inglese

1.3.1 Determinazione della lunghezza della chiave

Supponendo di avere un testo cifrato usando una chiave lunga n, significherebbe che le lettere posizionate a distanza n sono tutte cifrate con la stessa permutazione, quindi estraendole e concatenandole fra loro si otterrebbe una stringa cifrata interamente con un singolo cifrario a sostituzione semplice. Ora però bisogna distinguere due casi:

1. La frequenza delle lettere nella stringa è simile alla tabella 1.6 e quindi la supposizione sulla lunghezza della chiave è corretta

2. La frequenza delle lettere nella stringa è pressoché casuale quindi la supposizione è sbagliata.

Per capire in quale dei due casi ci si trova è necessario introdurre un particolare indice chiamato indice di coincidenza.

Definizione. *Sia $s = c_1c_2c_3 \dots c_n$ una stringa di n caratteri alfabetici. L'**indice di coincidenza** di s, denotato come $\mathrm{IndCo}(s)$, è la probabilità che due caratteri scelti a caso in s siano identici.[1]*

Proposizione 1.3.1. *Sia s una stringa di lunghezza n composta da lettere dell'alfabeto {a,b,...,z} che identifichiamo con i valori $i = 0,1,\ldots,25$. Sia F_i il numero di occorrenze della lettera i in s. Allora l'indice di coincidenza di s, indicato con $\mathrm{IndCo}(s)$, è definito come:*

$$\mathrm{IndCo}(s) = \frac{1}{n(n-1)} \sum_{i=0}^{25} F_i(F_i - 1) \tag{1.1}$$

Dimostrazione. Consideriamo la probabilità di estrarre due lettere identiche da s. Poiché s ha lunghezza n, il numero totale di modi per scegliere due lettere qualsiasi da s è

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

Per una lettera i, il numero di modi per scegliere due occorrenze di quella stessa lettera è

$$\binom{F_i}{2} = \frac{F_i(F_i-1)}{2}.$$

Sommando ora su tutte le lettere dell'alfabeto, il numero totale di coppie di lettere identiche corrisponde a

$$\sum_{i=0}^{25} \binom{F_i}{2} = \sum_{i=0}^{25} \frac{F_i(F_i-1)}{2}.$$

La probabilità di estrarre due lettere uguali è quindi il rapporto tra il numero di coppie di lettere identiche e il numero di tutte le possibili coppie di lettere

$$\frac{\sum_{i=0}^{25} \binom{F_i}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{25} \frac{F_i(F_i-1)}{2}}{\frac{n(n-1)}{2}} = \frac{\sum_{i=0}^{25} F_i(F_i-1)}{n(n-1)}.$$

Si può quindi definire l'indice di coincidenza $\mathrm{IndCo}(s)$ come

$$\mathrm{IndCo}(s) = \frac{1}{n(n-1)} \sum_{i=0}^{25} F_i(F_i-1).$$

□

Definizione. *Un pattern s di lunghezza $n \mod 26$ è random se*

$$\frac{F_i(s)}{n} = \frac{F_j(s)}{n} \qquad \forall i,j.$$

Proposizione 1.3.2. *Sia s un pattern di lunghezza $n = 26m$, $m \geq 1$.*

i) Se s è random allora

$$\mathrm{IndCo}(s) = \frac{m-1}{26m-1} \longrightarrow \frac{1}{26} \qquad per\ m \longrightarrow \infty$$

ii) $\mathrm{IndCo}(s)$ è minimo se e solo se s è random.

Dimostrazione. Sia s un pattern di lunghezza $n = 26m \mod 26$ tale che

$$\frac{F_i(s)}{n} = \frac{F_j(s)}{n} \qquad \forall i, j \in \{0, \ldots, 25\}.$$

Equivale a dire che tutte le frequenze $F_i(s)$ sono uguali, quindi esiste un valore t per cui $F_i(s) = t \ \ \forall\, i$.
Considerando che la somma di tutte le frequenze è n si ottiene

$$\sum_{i=0}^{25} F_i(s) = \sum_{i=0}^{25} t = 26t = n = 26m.$$

Da cui segue $t = m$. Quindi se s è random la frequenza di ogni lettera è pari ad m. Calcolando separatamente numeratore e denominatore dell'indice di coincidenza

$$\sum_{i=0}^{25} F_i(s)\,(F_i(s) - 1) = \sum_{i=0}^{25} m(m-1) = 26(m^2 - m) = 26m(m-1).$$

$$n(n-1) = 26m(26m-1).$$

Pertanto,

$$\mathrm{IndCo}(s) = \frac{26m(m-1)}{26m(26m-1)} = \frac{m-1}{26m-1}.$$

Studiandone il comportamento per $m \longrightarrow \infty$

$$\lim_{m\to\infty} \frac{m-1}{26m-1} = \frac{1}{26}.$$

Questo completa la dimostrazione della parte i).
Per dimostrare che $\mathrm{IndCo}(s)$ assume il valore minimo esattamente quando tutte le frequenze $F_i(s)$ sono uguali, osserviamo che

$$\sum_{i=0}^{25} F_i(s)\,(F_i(s) - 1) = \sum_{i=0}^{25} [F_i(s)^2 - F_i(s)] = \left(\sum_{i=0}^{25} F_i(s)^2\right) - \left(\sum_{i=0}^{25} F_i(s)\right).$$

Considerando che $\sum_{i=0}^{25} F_i(s) = n = 26m$ è costante, minimizzare $\sum_{i=0}^{25} F_i(s)\,(F_i(s)-1)$ equivale a minimizzare $\sum_{i=0}^{25} F_i(s)^2$.
Per una somma di valori positivi $F_i(s)$ con somma fissa, la somma dei quadrati è minima se e solo se tutti i valori $F_i(s)$ sono uguali. Ma $F_i(s)$ tutti uguali significa $F_i(s) = m$ per ogni $i = 0, \ldots, 25$ e quindi l'indice di coincidenza come nel punto i) risulta essere

$$\mathrm{IndCo}(s) = \frac{m-1}{26m-1}$$

che è anche il valore minimo. □

Supponiamo che s sia la stringa $s =$ "Unfortunately, no one can be told what the Matrix is." Ignorando spazi e punteggiatura, la tabella delle frequenze sarà così composta:

Lettera	a	b	c	d	e	f	h	i	l	m	n	o	r	s	t	u	w	x	y
i	0	1	2	3	4	5	7	8	11	12	13	14	17	18	19	20	22	23	24
F_i	4	1	1	1	4	1	2	2	2	1	5	4	2	1	6	2	1	1	1

Tabella 1.7: Frequenza delle lettere nella frase tratta da "The Matrix"

Applicando la formula (1.1):

$$\mathrm{IndCo}(s) = \frac{1}{42 \cdot 41}(4{\cdot}3+4{\cdot}3+2{\cdot}1+2{\cdot}1+2{\cdot}1+5{\cdot}4+4{\cdot}3+2{\cdot}1+6{\cdot}5+2{\cdot}1) \approx 0.0557 \quad (1.2)$$

Per capire se la stringa analizzata contiene lettere casuali è necessario confrontare il valore calcolato con quello che identifica un pattern random definito nella proposizione 1.3.2. Sebbene la differenza tra i valori $\frac{1}{26} \approx 0.0385$ e 0.0557 possa sembrare esigua, essa risulta comunque sufficiente per distinguere un testo in chiaro da uno cifrato con un cifrario a sostituzione semplice.
Supponiamo di aver intercettato un messaggio s e di voler verificare se la chiave di cifratura utilizzata abbia lunghezza k. È utile dividere s in k blocchi $s_1, s_2, \ldots, s_k$ dove ogni blocco s_i contiene le lettere di s nelle posizioni $i, i+k, i+2k, \ldots$:

$$s_i^k = c_i c_{i+k} c_{i+2k} \ldots \quad (1.3)$$

Se la lunghezza k ipotizzata è corretta, ciascun blocco s_i^k conterrà solo lettere cifrate con la stessa permutazione, il che significa che le loro frequenze saranno simili a quelle della lingua di riferimento.

Definizione. *Data una stringa s suddivisa in k blocchi s_i^k, la media degli indici di coincidenza* $\mathrm{IndCo}(s_i^k)$ $\forall i = 1, \ldots, k$ *è definita come:*

$$\overline{\mathrm{IndCo}}(s) = \frac{1}{k} \sum_{i=0}^{k} \mathrm{IndCo}(s_i^k). \tag{1.4}$$

Per verificare tale ipotesi, si calcola $\mathrm{IndCo}(s_i^k)$ per ogni blocco s_i^k e successivamente si confronta la media $\overline{\mathrm{IndCo}}(s)$ con l'indice di coincidenza della lingua interessata, che per quella inglese corrisponde a circa 0.065.
Se i due indici sono circa uguali (solitamente è sufficiente che $\overline{\mathrm{IndCo}}(s)$ sia maggiore di 0.06), allora l'ipotesi sulla lunghezza k della chiave può essere considerata corretta.

1.3.2 Individuazione della chiave nota la sua lunghezza

Il passo successivo richiede il confronto delle stringhe $s_1, \ldots, s_k$ con un'ulteriore stringa. Per farlo ci serviamo di un altro strumento, l'indice di coincidenza mutuo. L'idea alla base è abbastanza semplice: tutte le k stringhe sono cifrate utilizzando una diversa permutazione dell'alfabeto. Per esempio, la stringa s_i è traslata di β_i posizioni mentre s_j di β_j. Ci si aspetta che le frequenze delle lettere in s_i e s_j combacino al meglio quando viene applicata ad s_j un'ulteriore traslazione definita come

$$\sigma \equiv \beta_j - \beta_i \mod 26. \tag{1.5}$$

È utile fornire ora una definizione di indice di coincidenza mutuo.

Definizione. *Siano $s = c_1c_2c_3 \ldots c_n$ e $t = d_1d_2d_3 \ldots d_m$ stringhe di lettere dell'alfabeto. L'**indice di coincidenza mutuo** di s e t, denotato come* $\mathrm{MutIndCo}(s,t)$*, è la probabilità che un carattere scelto a caso da s e un carattere scelto a caso da t siano identici. [1]*

Proposizione 1.3.3. *Siano s e t due stringhe definite su un alfabeto di 26 simboli {a,b,...,z} che identifichiamo con i valori i = 0,1,...,25. Sia $F_i(s)$ il numero di occorrenze di i in s e $F_i(t)$ il numero di occorrenze di i in t. Se n ed m sono le lunghezze rispettivamente di s e t allora l'indice di coincidenza mutuo è definito come:*

$$\mathrm{MutIndCo}(s,t) = \frac{1}{nm} \sum_{i=0}^{25} F_i(s)F_i(t) \tag{1.6}$$

Dimostrazione. Sia u un indice in {1,...,n} e v un indice in {1,...,m}. Il numero totale di possibili coppie (u, v) è $n \cdot m$.
La probabilità che, scegliendo casualmente u, il simbolo in posizione u di s sia uguale ad i è $\frac{F_i(s)}{n}$.

Analogamente, scegliendo casualmente v, la probabilità che il simbolo in posizione v di t sia uguale ad i è $\frac{F_i(t)}{m}$.
Poiché la scelta di u e v è indipendente, la probabilità che il simbolo $u$esimo di s ed il simbolo $v$esimo di t siano entrambi uguali ad i è il prodotto tra le due probabilità:

$$\frac{F_i(s)}{n} \cdot \frac{F_i(t)}{m} = \frac{F_i(s)\, F_i(t)}{n\, m}$$

Sommando ora su tutti i simboli dell'alfabeto otteniamo

$$\sum_{i=0}^{25} \frac{F_i(s)\, F_i(t)}{n\, m} = \frac{1}{n\, m} \sum_{i=0}^{25} F_i(s)\, F_i(t).$$

□

Proposizione 1.3.4. *Il valore di* $\mathrm{MutIndCo}(s,t)$ *è massimo, al variare delle stringhe* s *e* t *di data lunghezza, quando le frequenze delle lettere di* s *e di* t *sono uguali.*

Dimostrazione. Siano le stringhe $s,\ t$ di lunghezze rispettive $n,\ m$ e siano

$$\alpha_i(s) = \frac{F_i(s)}{n}, \quad \alpha_i(t) = \frac{F_i(t)}{m} \qquad \forall i = 0, \ldots, 25.$$

Siano $\alpha(s)$ e $\alpha(t)$ due distribuzioni di probabilità

$$\alpha(s) = \begin{pmatrix} \alpha_0(s) \\ \vdots \\ \alpha_{25}(s) \end{pmatrix}, \qquad \alpha(t) = \begin{pmatrix} \alpha_0(t) \\ \vdots \\ \alpha_{25}(t) \end{pmatrix}$$

tali che

$$S = \sum_{i=0}^{25} \alpha_i(s)\alpha_i(t) = < \alpha_i(s), \alpha_i(t) >$$

dove $<\quad,\quad>$ è il prodotto scalare in $\mathbb{R}^{26}$.
Per Cauchy-Schwarz $< \alpha_i(s), \alpha_i(t) >$ è massimo se $\alpha(t) = \lambda\, \alpha(s)$ con $\lambda \geq 0$.
In tal caso

$$\underbrace{\sum_{i=0}^{25} \alpha_i(t)}_{1} = \lambda \underbrace{\sum_{i=0}^{25} \alpha_i(s)}_{1} = \lambda.$$

Da cui segue che $\lambda = 1$ quindi $\alpha(t) = \alpha(s)$. □

Le proprietà dell'indice di coincidenza mutuo permettono di confermare se la traslazione ipotizzata è corretta. Infatti, se le due stringhe s e t sono cifrate con la stessa traslazione (quindi con lo stesso cifrario a sostituzione semplice), il valore di $\mathrm{MutIndCo}(s,t)$ sarà più alto. In caso

contrario, quando le due stringhe sono cifrate con traslazioni diverse, il valore di $\mathrm{MutIndCo}(s,t)$ sarà più basso.
Tornando ora all'attacco al cifrario di Vigenère, dopo aver diviso il testo cifrato in k blocchi $s_1,\ldots,s_k$ ognuno dei quali è cifrato con una traslazione $\beta_1,\ldots,\beta_k$, l'obiettivo è quello di confrontare la stringa $i$esima con quella ottenuta traslando le lettere in s_j di un certo valore σ nell'alfabeto. Si confronta quindi s_i con $s_j+\sigma$. Supponendo di aver scelto il giusto valore di σ ($\sigma=\beta_i-\beta_j$) allora $s_j+\sigma$ è stato traslato di un valore pari a β_i dal messaggio in chiaro, il che significa che s_i e $s_j+\sigma$ sono stati cifrati utilizzando la stessa traslazione quindi avranno un indice di coincidenza mutuo piuttosto alto.
Si procede calcolando l'indice di coincidenza mutuo per tutte le coppie (s_i,s_j) possibili e per ogni valore di σ:

$$\mathrm{MutIndCo}(s_i,s_j+\sigma) \qquad \text{per} \quad 1\le i<j\le k \quad \text{e} \quad 0\le\sigma\le 25$$

Analizzando i risultati ottenuti, si sceglieranno i valori di $\mathrm{MutIndCo}(s_i,s_j+\sigma)$ maggiori di 0.065 e in quei casi molto probabilmente $\beta_i-\beta_j\equiv\sigma \mod 26$. Attraverso questo procedimento viene generato un sistema di equazione nelle variabili $\beta_1,\ldots,\beta_k$. Alcune di esse saranno fuorvianti ma dopo diversi tentativi verranno trovati dei valori $\gamma_2,...,\gamma_k$ tali che

$$\beta_2=\beta_1+\gamma_2,\quad \beta_3=\beta_1+\gamma_3,\quad \ldots,\quad \beta_k=\beta_1+\gamma_k.$$

A questo punto il trucco diventa piuttosto semplice, se la parola chiave inizia per A, allora la seconda lettera della chiave sarà A traslata di γ_2 posizioni, la terza A traslata di γ_3 e così via. Quindi tutto ciò che resta da fare è provare le 26 possibili lettere iniziali e decifrare il messaggio utilizzando le 26 parole chiave corrispondenti. Basterà analizzare i primi caratteri dei 26 ipotetici testi in chiaro trovati per capire quale sia quello corretto.

1.4 Crittoanalisi pratica del cifrario di Vigenère

In questa sezione verranno messi in pratica i passaggi per crittoanalizzare un testo cifrato con il cifrario di Vigenère che sono stati analizzati nella sezione 1.3. Il testo che verrà preso in esame compare nella Tabella 1.8.

1.4.1 Fase 1: trovare la lunghezza della chiave

Partiamo applicando il metodo Kasiski individuando nel testo le ripetizioni di lettere a gruppi di tre tenendo traccia delle relative posizioni. Per fare questo ci serviamo di uno script Python in modo da velocizzare il processo (Figura 1.1).

```python
import re
from collections import defaultdict
def find_trigrams(text):
    # Rimuove caratteri diversi dalle lettere e converte il testo in
    ↪ minuscolo
    clean_text = re.sub(r'[^a-zA-Z]', '', text)

    trigram_positions = defaultdict(list)

    # Estrae i trigrammi
    for i in range(len(clean_text) - 2):
        trigram = clean_text[i:i+3]
        trigram_positions[trigram].append(i)

    return {trigram: positions for trigram, positions in
    ↪ trigram_positions.items() if len(positions) > 1}

if __name__ == "__main__":
    text = ""   # Inserisci il testo qui
    trigrams = find_trigrams(text)

    for trigram, positions in trigrams.items():
        print(f"Trigram: '{trigram}' found at positions: {positions}")
```

Figura 1.1: Script Python per individuare le occorrenze di ogni trigramma e le relative posizioni

edxsh	mjrrj	bonov	jivuc	xbvrh	yzfbk	iyxyy	shnhb	hqsag	lhbqe
lrhew	fxyfl	aijre	tletu	wgmiu	dkmoz	huype	gfbsj	ydbvh	uiqge
kqqnr	tletu	wuely	awzsn	qjbam	wbrhq	staqg	aejco	gewyp	hcsxc
dnrtf	iomch	xjbem	zirqu	unbvu	bjyns	vzsrh	myhht	rglum	emzig
linou	mgrpv	yvbls	axjba	mizyq	uhakm	qeiqn	ewmeh	ebcht	dsnhh
yafbv	nxehe	wimbr	jbekm	rumbu	fshnu	rshai	tbvrw	ehshn	tbvcy
rltoi	iiunw	bvrwe	hshnt	bvcyt	ltoii	eqnxz	gjmbf	bxipy	ijisb
brbad	nozmh	uihut	mpsge	rfehn	pesjb	ekpcb	h		

Tabella 1.8: Testo cifrato con il cifrario di Vigenère

La lista di trigrammi con la loro posizione nel testo cifrato e la relativa distanza è mostrata nella Tabella 1.9.

Attraverso il metodo Kasiski possiamo vedere che la maggior parte delle distanze tra i trigrammi è divisibile per 8, quindi la lunghezza della chiave probabilmente sarà questa. Proviamo comunque ad applicare l'indice di coincidenza attraverso uno script Python (Figura 1.2).

Lunghezza della chiave	Indice medio	Indici di coincidenza
4	0.056	0.060, 0.052, 0.045, 0.069
5	0.043	0.033, 0.040, 0.048, 0.044, 0.050
6	0.048	0.037, 0.056, 0.046, 0.050, 0.062, 0.038
7	0.050	0.070, 0.054, 0.038, 0.061, 0.047, 0.047, 0.034
8	0.075	0.076, 0.073, 0.080, 0.092, 0.070, 0.065, 0.044, 0.101
9	0.044	0.040, 0.049, 0.032, 0.037, 0.051, 0.037, 0.056, 0.045, 0.051

Tabella 1.10: Indici di coincidenza per diverse lunghezze della chiave

La tabella degli indici di coincidenza conferma che il valore trovato con il metodo Kasiski è corretto, quindi la chiave avrà lunghezza 8 (Tabella 1.10).

1.4.2 Fase 2: individuare la chiave

Ora è necessario dividere il testo in otto blocchi prendendo una lettera ogni 8:

$$s_1 = \text{erihxhqxemhsireqheprxqjhllpxqiehxrmswvirbijybugeb}$$
$$s_2 = \text{djvyyqeytiujqtljqjhtjuymuivjuqbhejbheciwvimiaiesh}$$
$$s_3 = \text{xbuzyslfluyyglybsccfbunymnybhncyhbfahyiecebjdhrj}$$
$$s_4 = \text{socfsarledpddeeaatosiensheovaaehaeesisruhyqfinufb}$$
$$s_5 = \text{hnxbhghatkebktwmagxombvhmubmkwtfwkhthlnsrnbsotee}$$
$$s_6 = \text{mobknleiumgvquzwqecmzvztzmlimmdbimnbntwhlxxbzmhk}$$
$$s_7 = \text{jvvihhwjwofhqwsbgwdciusrigszqesvmruvtobntzibmpnp}$$
$$s_8 = \text{rjrybbfrgzbununraynhrbrggrayehnnburrbivtogprhspc}$$

Trigramma	Posizioni	Differenza
bam	122, 218	$96 = 2^5 \cdot 3$
bek	266, 378	$112 = 2^4 \cdot 7$
bvc	295, 319	$24 = 2^3 \cdot 3$
bvr	21, 285, 309	$264 = 2^3 \cdot 3 \cdot 11$, $24 = 2^3 \cdot 3$
emz	163, 195	$32 = 2^5$
ehe	238, 257	$19 = 19$
ehs	289, 313	$24 = 2^3 \cdot 3$
etu	67, 107	$40 = 2^3 \cdot 5$
hew	52, 258	$206 = 2 \cdot 103$
hqs	40, 128	$88 = 2^3 \cdot 11$
hnt	292, 316	$24 = 2^3 \cdot 3$
hsh	290, 314	$24 = 2^3 \cdot 3$
hui	94, 358	$264 = 2^3 \cdot 3 \cdot 11$
jba	121, 217	$96 = 2^5 \cdot 3$
jbe	161, 265, 377	$104 = 2^3 \cdot 13$, $112 = 2^4 \cdot 7$
let	66, 106	$40 = 2^3 \cdot 5$
lto	300, 324	$24 = 2^3 \cdot 3$
mbf	272, 336	$64 = 2^6$
mzi	164, 196	$32 = 2^5$
nrt	103, 151	$48 = 2^4 \cdot 3$
ntb	293, 317	$24 = 2^3 \cdot 3$
oii	302, 326	$24 = 2^3 \cdot 3$
rjb	8, 264	$256 = 2^8$
rlt	299, 323	$24 = 2^3 \cdot 3$
rwe	287, 311	$24 = 2^3 \cdot 3$
shn	35, 275, 291, 315	$240 = 2^4 \cdot 3 \cdot 5$, $16 = 2^4$, $24 = 2^3 \cdot 3$
tbv	284, 294, 318	$10 = 2 \cdot 5$, $24 = 2^3 \cdot 3$
tle	65, 105	$40 = 2^3 \cdot 5$
toi	301, 325	$24 = 2^3 \cdot 3$
tuw	68, 108	$40 = 2^3 \cdot 5$
vcy	296, 320	$24 = 2^3 \cdot 3$
vrw	286, 310	$24 = 2^3 \cdot 3$
weh	288, 312	$24 = 2^3 \cdot 3$
xjb	160, 216	$56 = 2^3 \cdot 7$
yrl	298, 322	$24 = 2^3 \cdot 3$
zhe	27, 48	$21 = 3 \cdot 7$

Tabella 1.9: Trigrammi ripetuti nel testo cifrato (Tabella 1.8)

```python
    from collections import Counter

def index_of_coincidence(text: str, key_length: int):
    blocks = ['' for _ in range(key_length)]

    # Divide il testo in blocchi di lunghezza pari alla lunghezza della
    ↪  chiave
    for i, char in enumerate(text):
        blocks[i % key_length] += char

    ic_values = []

    # Calcola l'indice di coincidenza per ogni blocco
    for block in blocks:
        n = len(block)
        if n < 2:
            continue  # Evita le divisioni per zero

        freq_counts = Counter(block)  # Conta le occorrenze di ogni lettera
        numerator = sum(f * (f - 1) for f in freq_counts.values())
        ic = numerator / (n * (n - 1))

        if ic != 0:
            ic_values.append(ic)

    mean_ic = sum(ic_values) / len(ic_values) if ic_values else 0

    return ic_values, mean_ic

text = ""   # Inserisci il testo qui
key_length = 4  # Inserisci la lunghezza della chiave ipotizzata
print(index_of_coincidence(text, key_length))
```

Figura 1.2: Script Python per calcolare gli indici di coincidenza

A questo punto bisogna calcolare l'indice di coincidenza mutuo fra i vari blocchi in modo da confrontare il blocco s_i con il blocco $s_j + \sigma$ per $\sigma = 0, \ldots, 25$:

$$\text{MutIndCo}(s_i, s_j + \sigma) \qquad \text{per} \quad 1 \leq i < j \leq 8 \quad \text{e} \quad 0 \leq \sigma \leq 25$$

Per farlo useremo lo script Python rappresentato in Figura 1.3. I risultati ottenuti sono mostrati nella Tabella 1.11.

```python
import string
import json
from collections import Counter

def shift_text(text, sigma):
    """
    Trasla ogni lettera del testo di sigma posizioni.
    Si assume che il testo contenga lettere minuscole.
    """
    shifted = []
    for ch in text:
        if ch in string.ascii_lowercase:
            # Calcola la nuova lettera: (posizione corrente + sigma) modulo
            ↪ 26
            new_ch = chr((ord(ch) - ord('a') + sigma) % 26 + ord('a'))
            shifted.append(new_ch)
        else:
            # Se il carattere non è una lettera, lo lasciamo inalterato
            shifted.append(ch)
    return ''.join(shifted)

def mutual_index_of_coincidence(text1, text2):
    """
    Calcola l'indice di coincidenza mutuo tra text1 e text2.
    La formula è:
        MutIndCo(s, t) = (sum_{i=1}^{26} F_i(s)*F_i(t)) / (|s| * |t|)
    dove F_i(s) è il numero di occorrenze della i-esima lettera (a-z) in s.
    """
    n = len(text1)
    m = len(text2)
    counter1 = Counter(text1)
    counter2 = Counter(text2)
    total = 0
    for letter in string.ascii_lowercase:
        total += counter1.get(letter, 0) * counter2.get(letter, 0)
    return total / (n * m)
```

```python
# Gli 8 blocchi di testo trovati in precedenza
blocks = {
    1: "erihxhqxemhsireqheprxqjhllpxqiehxrmswvirbijybugeb",
    2: "djvyyqeytiujqtljqjhtjuymuivjuqbhejbheciwvimiaiesh",
    3: "xbuzyslfluyyglybsccfbunymnybhncyhbfahyiecebjdhrj",
    4: "socfsarledpddeeaatosiensheovaaehaeesisruhyqfinufb",
    5: "hnxbhghatkebktwmagxombvhmubmkwtfwkhthlnsrnbsotee",
    6: "mobknleiumgvquzwqecmzvztzmlimmdbimnbntwhlxxbzmhk",
    7: "jvvihhwjwofhqwsbgwdciusrigszqesvmruvtobntzibmpnp",
    8: "rjrybbfrgzbununraynhrbrggrayehnnburrbivtogprhspc"
}

# Calcoliamo MutIndCo(s_i, s_j + sigma) per ogni coppia (i,j) con 1<=i<j<=8 e
↪  per sigma da 0 a 25
results = {}

for i in range(1, 9):
    for j in range(i+1, 9):
        sigma_results = {}
        for sigma in range(26):
            shifted_block = shift_text(blocks[j], sigma)
            mi = mutual_index_of_coincidence(blocks[i], shifted_block)
            sigma_results[sigma] = mi
        # Uso come chiave una stringa "i,j"
        key = f"{i},{j}"
        results[key] = sigma_results

# Esportiamo il dizionario in un file JSON
output_filename = "mutindco_results.json"
with open(output_filename, "w") as outfile:
    json.dump(results, outfile, indent=4)

print(f"Risultati esportati in '{output_filename}'")
```

Figura 1.3: Script Python per calcolare l'indice di coincidenza mutuo

i	j	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	2	0.056	0.032	0.027	0.044	0.036	0.024	0.034	0.042	0.046	0.047	0.036	0.036	0.030
1	3	0.041	0.022	0.032	0.054	0.032	0.038	0.061	0.038	0.027	0.045	0.060	0.033	0.028
1	4	0.049	0.035	0.031	0.052	0.061	0.036	0.027	0.038	0.036	0.040	0.028	0.026	0.050
1	5	0.045	0.038	0.029	0.043	0.048	0.038	0.034	0.037	0.037	0.027	0.043	0.057	0.032
1	6	0.042	0.029	0.028	0.039	0.043	0.054	0.034	0.036	0.041	0.037	0.040	0.047	0.033
1	7	0.044	0.041	0.041	0.040	0.030	0.039	0.030	0.027	0.043	0.045	0.034	0.037	0.044
1	8	0.049	0.035	0.033	0.053	0.031	0.026	0.055	0.035	0.028	0.033	0.060	0.028	0.020
2	3	0.048	0.033	0.040	0.044	0.034	0.031	0.048	0.049	0.037	0.040	0.047	0.051	0.023
2	4	0.043	0.042	0.039	0.045	0.060	0.047	0.037	0.035	0.042	0.040	0.020	0.026	0.042
2	5	0.036	0.042	0.051	0.037	0.031	0.033	0.033	0.041	0.048	0.036	0.028	0.040	0.044
2	6	0.040	0.035	0.027	0.029	0.036	0.038	0.031	0.040	0.060	0.042	0.041	0.031	0.044
2	7	0.046	0.044	0.045	0.043	0.032	0.033	0.036	0.034	0.043	0.032	0.028	0.033	0.048
2	8	0.032	0.037	0.047	0.061	0.034	0.025	0.036	0.056	0.037	0.034	0.039	0.037	0.022
3	4	0.041	0.040	0.037	0.036	0.037	0.034	0.045	0.045	0.028	0.047	0.040	0.034	0.026
3	5	0.040	0.051	0.032	0.026	0.034	0.047	0.045	0.033	0.038	0.028	0.038	0.039	0.040
3	6	0.036	0.044	0.035	0.035	0.034	0.039	0.039	0.029	0.047	0.023	0.033	0.034	0.060
3	7	0.033	0.027	0.040	0.039	0.033	0.042	0.053	0.034	0.037	0.040	0.043	0.042	0.048
3	8	0.055	0.039	0.021	0.044	0.037	0.030	**0.070**	0.025	0.030	0.051	0.056	0.033	0.033
4	5	0.041	0.036	0.025	0.041	0.044	0.024	0.035	0.055	0.045	0.026	0.032	0.050	0.040
4	6	0.026	0.043	0.034	0.039	0.037	0.048	0.054	0.043	0.039	0.032	0.038	0.028	0.028
4	7	0.039	0.036	0.032	0.031	0.034	0.039	0.035	0.029	0.045	0.046	0.041	0.043	0.054
4	8	0.040	0.053	0.037	0.046	0.034	0.033	0.037	0.036	0.020	0.048	0.033	0.035	0.043
5	6	0.051	0.035	0.041	0.033	0.026	0.036	0.047	0.043	0.042	0.032	0.037	0.053	0.029
5	7	0.042	0.039	0.035	0.032	0.038	0.049	0.044	0.027	0.032	0.036	0.033	0.049	0.041
5	8	0.044	0.037	0.040	0.036	0.031	0.043	0.061	0.030	0.022	0.042	0.043	0.026	0.041
6	7	0.043	0.030	0.030	0.042	0.046	0.047	0.040	0.040	0.034	0.036	0.030	0.039	0.038
6	8	0.031	0.031	0.030	0.039	0.032	0.043	0.050	0.037	0.045	0.024	0.051	0.036	0.046
7	8	0.043	0.054	0.040	0.036	0.038	0.046	0.032	0.039	0.044	0.030	0.035	0.029	0.038

i	j	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
1	2	0.040	0.060	0.040	0.030	0.027	0.038	0.036	0.020	0.040	0.048	0.043	0.036	0.052
1	3	0.034	0.041	0.037	0.041	0.036	0.036	0.042	0.032	0.033	0.043	0.038	0.028	0.048
1	4	0.045	0.034	0.042	0.053	0.041	0.028	0.045	0.037	0.019	0.029	0.048	0.028	0.039
1	5	0.033	0.034	0.040	0.050	0.035	0.025	0.030	0.048	0.036	0.039	0.057	0.037	0.027
1	6	0.032	0.034	0.037	0.033	0.035	0.043	0.035	0.035	0.049	0.060	0.041	0.034	0.029
1	7	0.037	0.037	0.048	0.047	0.037	0.031	0.032	0.035	0.044	0.045	0.031	0.037	0.043
1	8	0.046	0.032	0.037	0.059	**0.063**	0.032	0.025	0.037	0.038	0.034	0.030	0.037	0.041
2	3	0.031	0.037	0.041	0.025	0.039	0.043	0.034	0.031	0.036	0.052	0.044	0.031	0.030
2	4	0.030	0.029	0.042	**0.065**	0.052	0.028	0.036	0.049	0.035	0.030	0.024	0.036	0.025
2	5	0.033	0.040	0.047	0.044	0.037	0.027	0.031	0.045	0.033	0.033	0.061	0.037	0.033
2	6	0.037	0.037	0.035	0.029	0.037	0.035	0.033	0.033	0.044	0.058	0.060	0.029	0.040
2	7	0.051	0.040	0.042	0.042	0.044	0.034	0.033	0.040	0.039	0.033	0.032	0.031	0.041
2	8	0.043	0.035	0.040	0.037	0.052	0.060	0.027	0.043	0.044	0.036	0.027	0.022	0.039
3	4	0.046	0.029	0.030	0.039	0.037	0.022	0.043	**0.064**	0.035	0.029	0.047	0.057	0.033
3	5	0.036	0.047	0.032	0.035	0.054	0.036	0.033	0.052	0.039	0.024	0.051	0.039	0.030
3	6	0.032	0.042	0.040	0.050	0.041	0.032	0.042	0.036	0.043	0.027	0.047	0.028	0.051
3	7	0.034	0.032	0.041	0.043	0.041	0.038	0.044	0.035	0.038	0.033	0.037	0.032	0.040
3	8	0.036	0.042	0.029	0.035	0.035	0.045	0.032	0.043	0.028	0.043	0.036	0.034	0.027
4	5	0.037	0.041	0.024	0.036	0.045	0.036	0.042	0.043	0.037	0.038	0.045	0.036	0.043
4	6	0.028	0.050	0.032	0.031	0.047	0.060	0.053	0.032	0.038	0.044	0.035	0.026	0.034
4	7	0.042	0.034	0.033	0.031	0.037	0.039	0.036	0.027	0.036	0.051	0.044	0.037	0.045
4	8	**0.074**	0.037	0.024	0.041	0.056	0.026	0.028	0.043	0.023	0.030	0.040	0.041	0.042
5	6	0.034	0.039	0.044	0.030	0.034	0.038	0.038	0.047	0.040	0.033	0.037	0.039	0.041
5	7	0.035	0.040	0.042	0.033	0.037	0.049	0.044	0.038	0.035	0.032	0.038	0.043	0.040
5	8	0.053	0.035	0.030	0.049	0.028	0.038	0.046	0.043	0.050	0.031	0.035	0.027	0.038
6	7	0.038	0.038	0.036	0.043	0.047	0.040	0.041	0.046	0.041	0.034	0.030	0.041	0.034
6	8	0.036	0.037	0.030	0.030	0.036	0.035	0.043	0.042	**0.069**	0.036	0.032	0.036	0.045
7	8	0.034	0.044	0.042	0.041	0.044	0.036	0.032	0.041	0.049	0.031	0.030	0.036	0.033

Tabella 1.11: Risultati $\mathrm{MutIndCo}(s_i, s_j + \sigma)$

Nelle tabelle sono stati evidenziati i valori superiori a 0.063 che corrispondo ai valori per cui è probabile che $\beta_i - \beta_j = \sigma$.

i	j	**MutIndCo**	**Spostamento**
1	8	0.063	$\beta_1 - \beta_8 = 17$
3	4	0.064	$\beta_3 - \beta_4 = 20$
3	8	0.070	$\beta_3 - \beta_8 = 6$
2	4	0.065	$\beta_2 - \beta_4 = 16$
4	8	0.074	$\beta_4 - \beta_8 = 13$
6	8	0.069	$\beta_6 - \beta_8 = 21$

Tabella 1.12: Valori di $\mathrm{MutIndCo} \geq 0.063$

Il prossimo passo sarà risolvere il sistema di equazioni che compaiono nell'ultima colonna della Tabella 1.12. Ci sono 6 equazioni per 6 variabili $\beta_1, \beta_2, \beta_3, \beta_4, \beta_6, \beta_8$.

$$\beta_1 = \beta_8 + 17, \quad \beta_3 = \beta_8 + 7, \quad \beta_3 = \beta_8 + 6, \quad \beta_4 = \beta_8 + 13, \quad \beta_2 = \beta_8 + 3, \quad \beta_6 = \beta_8 + 21.$$

In questo caso abbiamo due equazioni che portano a risultati diversi per β_3 quindi cerchiamo un altro valore alto di $\mathrm{MutIndCo}$ per poter scegliere quello corretto: $\beta_3 - \beta_6 = 12$ quindi $\beta_3 = \beta_8 + 7$. Per le restanti variabili che non compaiono nella tabella (β_5, β_7) andiamo a prendere i valori più alti nei rispettivi blocchi dalla Tabella 1.11.

$$\beta_5 - \beta_8 = 6, \quad \beta_7 - \beta_8 = 1.$$

Sostituendo con le equazioni precedenti:

$$\beta_5 = \beta_8 + 6, \quad \beta_7 = \beta_8 + 1.$$

Per semplicità, come blocco di riferimento terremo in considerazione s_8, in quanto tutte le equazioni sono in funzione di β_8. Per quanto il blocco s_8 venga ruotato, gli altri blocchi saranno anch'essi ruotati di una quantità fissa che è descritta nelle equazioni appena trovate:

- s_1 sarà ruotato di 17 posizioni rispetto a s_8
- s_2 sarà ruotato di 3 posizioni rispetto a s_8
- s_3 sarà ruotato di 7 posizioni rispetto a s_8
- s_4 sarà ruotato di 13 posizioni rispetto a s_8
- s_5 sarà ruotato di 6 posizioni rispetto a s_8
- s_6 sarà ruotato di 21 posizioni rispetto a s_8

- s_7 sarà ruotato di 1 posizione rispetto a s_8

Provando una ad una le 26 possibili chiavi (di cui alla Tabella 1.13) si nota come l'unica con la quale appare un testo in chiaro comprensibile è "EQUATION".

Rotazione	**Chiave**	**Testo decifrato**
0	RDHNGVBA	naqfbriragubhtujrsnprgurqvssvphygvrfbsgbqnlnaqgbzbeeb
1	SEIOHWCB	mzpeaqhqzftagstiqrmoqftqpurruogxfuqearfapmkmzpfayadda
2	TFJPIXDC	lyodzpgpyeszfrshpqlnpespotqqtnfwetpdzqezoljlyoezxzccz
3	UGKQJYED	kxncyofoxdryeqrgopkmodronsppsmevdsocypdynkikxndywybby
4	VHLRKZFE	jwmbxnenwcqxdpqfnojlncqnmroorlducrnbxocxmjhjwmcxvxaax
⋮	⋮	⋮
12	DPTZSHNM	boetpfwfouipvhixfgbdfuifejggjdvmujftpgupebzboeupnpssp
13	EQUATION	andsoeventhoughwefacethedifficultiesoftodayandtomorro
14	FRVBUJPO	zmcrndudmsgntfgvdezbdsgdcheehbtkshdrnesnczxzmcsnlnqqn
⋮	⋮	⋮
24	PBFLETZY	pcshdtktciwdjvwltuprtiwtsxuuxrjaixthduidspnpcsidbdggd
25	QCGMFUAZ	obrgcsjsbhvciuvkstoqshvsrwttwqizhwsgcthcromobrhcacffc

Tabella 1.13: Testi decifrati a partire da 1.8

È sufficiente continuare la decifrazione del testo usando la chiave individuata e aggiungere spazi ed eventuale punteggiatura dove necessario per ottenere l'intero messaggio:

> And so even though we face the difficulties of today and tomorrow, I still have a dream. It is a dream deeply rooted in the American dream. I have a dream that one day this nation will rise up and live out the true meaning of its creed: "We hold these truths to be self-evident, that all men are created equal. I have a dream that one day on the red hills of Georgia, the sons of former slaves and the sons of former slave owners will be able to sit down together at the table of brotherhood.[3]

[3] *I Have a Dream*, 1963, Martin Luther King

Capitolo 2

Algoritmi di collisione e attacchi Meet-in-the-Middle

Il rapporto tra i principi della teoria dell'Informazione formulata da Shannon e le problematiche della crittografia moderna è piuttosto evidente, ad esempio, nell'analisi delle collisioni e negli attacchi Meet-in-the-Middle (MITM).
Le collisioni, definite come il fenomeno per cui due input distinti generano lo stesso output, sottolineano l'importanza del calcolo probabilistico, concetto chiave della visione shannoniana della comunicazione. Gli attacchi MITM, invece, sfruttano simmetrie e vulnerabilità strutturali dei sistemi di cifratura a più livelli, riducendo significativamente il numero di operazioni necessarie per comprometterne la sicurezza.
Per arrivare a comprendere meglio algoritmi di collisione e attacchi Meet-in-the-Middle, è necessario soffermarsi sul ragionamento che ne sta alla base: in un contesto di ricerca, solitamente è molto più semplice trovare oggetti che corrispondono (collidono) piuttosto che cercare un particolare oggetto.

2.1 Il paradosso del compleanno

Il paradosso del compleanno, pur non essendo un vero e proprio paradosso matematico, è un buon metodo per illustrare il ragionamento su cui si basano gli algoritmi di collisione. Inoltre, il metodo di risoluzione di questo problema ricorda molto quello enunciato nel teorema che verrà presentato nella Sezione 2.2. Questo problema emerge dalla seguente domanda:

> Qual è la probabilità che, in un gruppo di n persone, almeno due di esse festeggino il compleanno nello stesso giorno?

Per rispondere a questa domanda si assume che un anno abbia 365 giorni (ignorando eventuali anni bisestili), che i compleanni siano indipendenti da persona a persona e che le possibi-

li date siano equiprobabili. Con queste premesse, per risolvere il paradosso del compleanno consideriamo la probabilità che tutte le persone del gruppo abbiano compleanni diversi.

$$\begin{aligned}\Pr\left(\begin{array}{c}\text{due persone hanno}\\ \text{lo stesso compleanno}\end{array}\right) &= 1 - \Pr\left(\begin{array}{c}\text{tutte le persone hanno}\\ \text{compleanni diversi}\end{array}\right)\\ &= 1 - \left(\frac{365!}{365^{30}\,(365-30)!}\right)\\ &\approx 70.6\%.\end{aligned}$$

Nel grafico in Figura 2.1 è possibile vedere i valori di probabilità relativi al paradosso del compleanno per gruppi di $2 \leq k \leq 70$ persone, in particolare per un gruppo di 23 persone la percentuale è circa del 50%.

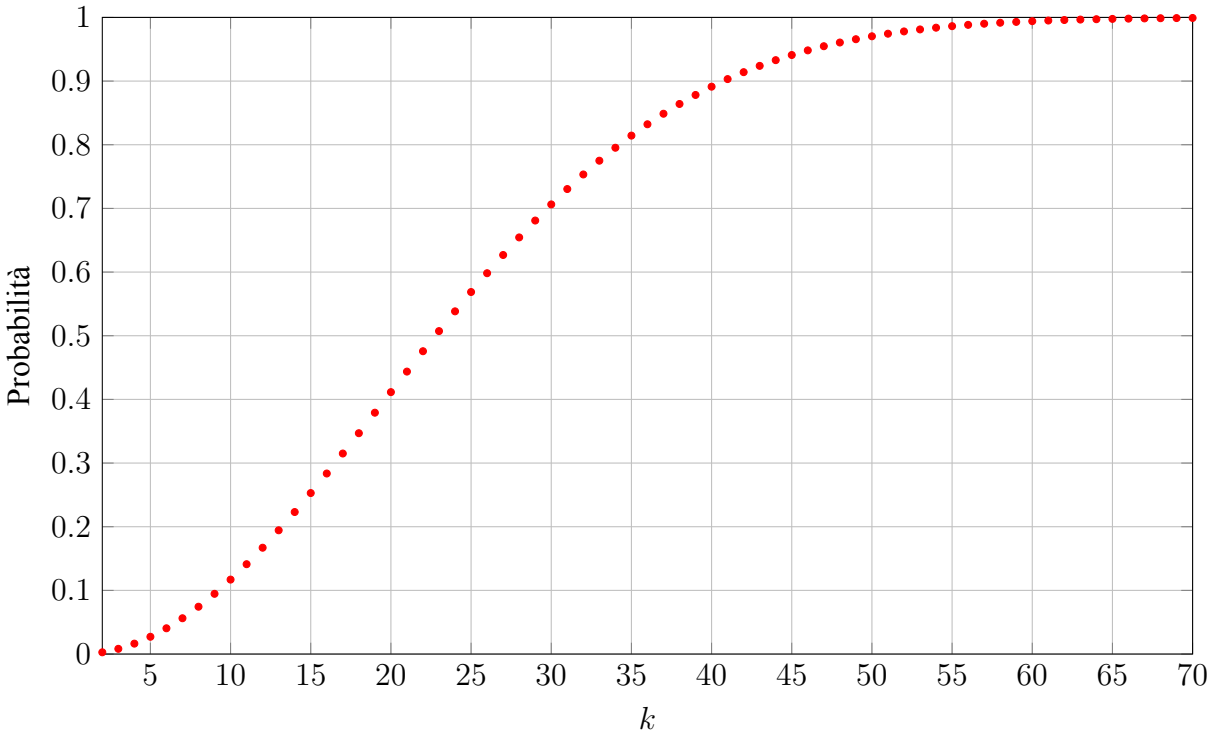

Figura 2.1: Probabilità di almeno una collisione per $2 \leq k \leq 70$ nel problema del compleanno.

2.2 Teorema della collisione

Si consideri un'urna contenente N elementi distinti. Bob ne estrae n e li inserisce in una lista; successivamente, estrae m elementi non necessariamente distinti per formare una seconda lista. Se n ed m sono entrambi leggermente maggiori di $\sqrt{N}$, allora la probabilità che nelle due liste ci sia almeno un elemento in comune è estremamente alta. Tale fenomeno evidenzia come il

numero di possibili collisioni cresca rapidamente all'aumentare della dimensione delle liste, nonostante lo spazio totale degli elementi sia molto ampio.
Premessa: Consideriamo un'urna con N palline, di cui n sono rosse e $N-n$ sono blu. In m estrazioni con reimmissione di una pallina alla volta, qual è la probabilità che almeno una pallina sia rossa?

Teorema 2.2.1 (Teorema della Collisione)**.**
Sia $\begin{cases} X = R \cup B, & \text{con } |X| = N,\ |R| = n,\ |B| = N-n \\ R \cap B = \emptyset \end{cases}$
$\Omega = X^m$ *con probabilità uniforme* $\underline{\Pr}$.
Se $E = \{(x_1, \ldots, x_m) : \exists\, i \in \{1, \ldots, m\} : x_i \in R\}$ *allora*

a)

$$\Pr(E) = 1 - \left(1 - \frac{n}{N}\right)^m \tag{2.1}$$

b)

$$\Pr(E) \geq 1 - e^{-\frac{mn}{N}} \tag{2.2}$$

Dimostrazione.

a) Si osservi che $\Omega|E = \underbrace{B \times \cdots \times B}_{m \text{ volte}}$. Pertanto

$$\begin{aligned} \Pr(E) &= 1 - \Pr(\Omega|E) \\ &= 1 - \left(\frac{|B|}{N}\right)^m \\ &= 1 - \left(\frac{N-n}{N}\right)^m. \end{aligned}$$

Questo conclude la dimostrazione del punto a).

b) Si utilizza la diseguaglianza

$$e^{-x} \geq 1 - x \qquad \forall x \in \mathbb{R}.$$

Ponendo $x = \frac{n}{N}$ si ottiene:

$$e^{-\frac{n}{N}} \geq 1 - \frac{n}{N}.$$

Ora, elevando entrambi i membri alla potenza di m:

$$\left(e^{-\frac{n}{N}}\right)^m = e^{-\frac{mn}{N}} \geq \left(1 - \frac{n}{N}\right)^m.$$

Aggiungendo 1 ad entrambi i membri ed invertendo il segno:

$$1 - e^{-\frac{mn}{N}} \leq 1 - \left(1 - \frac{n}{N}\right)^m.$$

Ma da quanto visto al punto a):

$$\Pr(E) = 1 - \left(1 - \frac{n}{N}\right)^m.$$

Pertanto:

$$\Pr(E) \geq 1 - e^{-\frac{mn}{N}}.$$

Questo conclude la dimostrazione del punto b).

□

Per collegare il Teorema 2.2.1 con la ricerca di corrispondenze tra due liste di dati, tipica degli algoritmi di collisione, è particolarmente utile immaginare la lista di numeri come un'urna contenente N palline blu numerate. Dopo aver creato la prima lista di n palline numerate distinte, queste vengono marcate di rosso e reinserite nell'urna. La seconda lista viene formata estraendo m palline una alla volta, annotando numero e colore e poi reinserendole nell'urna. Così facendo, la probabilità di estrarre almeno una pallina rossa corrisponde alla probabilità che le due liste di dati, rispettivamente di n ed m elementi, abbiano almeno un elemento in comune.

Esempio 2.2.1. Si consideri un insieme contenente N oggetti. Bob ne sceglie n con i quali crea una lista, li rimette nell'insieme iniziale e successivamente ne estrae altri n ognuno con reinserimento. Quanto deve essere grande n per ottenere una probabilità di collisione prima del 50% e poi del 99.99%?
È sufficiente utilizzare il limite inferiore definito dalla formula (2.2).

$$\Pr(\text{collisione}) \geq 1 - e^{-\frac{n^2}{N}} = \frac{1}{2}. \tag{2.3}$$

$$e^{-\frac{n^2}{N}} = \frac{1}{2} \quad \Longrightarrow \quad -\frac{n^2}{N} = \ln\frac{1}{2} \quad \Longrightarrow \quad n = \sqrt{N \cdot \ln 2} \approx 0.83\sqrt{N}.$$

$$\Pr(\text{collisione}) \geq 1 - e^{-\frac{n^2}{N}} = 0.9999 = 1 - 10^{-4}. \tag{2.4}$$

Analogamente,

$$e^{-\frac{n^2}{N}} = 10^{-4} \quad \Longrightarrow \quad -\frac{n^2}{N} = \ln 10^{-4} \quad \Longrightarrow \quad n = \sqrt{N \cdot \ln 10^4} \approx 3.035\sqrt{N}.$$

Quindi, per avere una probabilità di collisione del 50% è sufficiente che le liste abbiano una lunghezza n leggermente inferiore a $\sqrt{N}$, mentre per arrivare a 99.99%, n deve essere leggermente maggiore del triplo di $\sqrt{N}$.

2.3 Un algoritmo di collisione per il logaritmo discreto

In crittografia, gli algoritmi di collisione vengono spesso sfruttati per risolvere i problemi matematici difficili che stanno alla base dei sistemi crittografici a chiave pubblica.
Nel caso del problema del logaritmo discreto (DLP), se considerato in un campo finito $\mathbb{F}_p$, è possibile sviluppare un algoritmo di collisione randomizzato che richiede in media $\sqrt{p}$ passi per trovare la soluzione. L'algoritmo analizzato in questa sezione viene fornito con il solo scopo di poter applicare il Teorema di Collisione 2.2.1. Esiste infatti l'algoritmo *Baby step - Giant step* che trova il logaritmo discreto in $2\sqrt{p}$ passi, con il vantaggio che in assenza di collisioni non ha soluzione.

Proposizione 2.3.1. *Sia G un gruppo se sia $g \in G$ un elemento di ordine N, ossia $g^N = e$ e nessuna potenza di g con esponente minore di N è uguale ad e. Assumendo che il problema del logaritmo discreto*

$$g^x = h \tag{2.5}$$

abbia soluzione, allora questa può essere trovata in $\mathcal{O}(\sqrt{N})$ passi, dove ogni passo corrisponde a un'operazione di elevamento a potenza nel gruppo G.

Dimostrazione. È conveniente esprimere x come differenza tra esponenti $x = y - z$, quindi (2.5) diventa:

$$g^{y-z} = h \quad \Longrightarrow \quad \frac{g^y}{g^z} = h \quad \Longrightarrow \quad g^y = h \cdot g^z.$$

Per cercare una soluzione costruiamo due liste, una con i valori di g^y e l'altra con quelli di $h \cdot g^z$. Formiamo la prima lista scegliendo casualmente n esponenti $y_1, y_2, \ldots, y_n$ compresi tra 1 ed N e calcoliamo i valori $g^{y_1}, g^{y_2}, \ldots, g^{y_n}$ in G. Questi valori appartengono all'insieme $S = \{1, g, g^2, \ldots, g^{N-1}\}$ che, facendo un'analogia con il Teorema 2.2.1, può essere visto come un urna contenente N palline numerate mentre la scelta degli n esponenti è simile alla colorazione delle palline estratte.
Per la seconda lista selezioniamo altri n esponenti casuali $z_1, z_2, \ldots, z_n$ e calcoliamo i valori $h \cdot g^{z_1}, h \cdot g^{z_2}, \ldots, h \cdot g^{z_n}$ in G.
Dall'ipotesi che (2.5) ha soluzione e che h è una qualche potenza di g, segue che anche i valori $h \cdot g^{z_i}$ appartengono all'insieme S.
Per assicurarsi di trovare un elemento comune tra le due liste è necessario scegliere un valore

opportuno di n. Servendosi del limite inferiore del teorema della collisione (Teorema 2.2.1), si nota che scegliendo per esempio $n \approx 3\sqrt{N}$ la probabilità di collisione è maggiore di 99.98% (se è richiesta una probabilità più alta è sufficiente aumentare la costante che moltiplica $\sqrt{N}$). Una volta trovata una collisione, cioè $g^y = h \cdot g^z$, si può concludere che $x = y - z$ è soluzione del DLP (nel caso in cui x risulta essere negativo si può sempre utilizzare $x = y - z + N$ dal momento che $g^N = 1$).
Analizzando il tempo di calcolo, si nota che la creazione di ciascuna lista richiede n calcoli di g^i per $i \in [1, N]$ ognuno dei quali impiega mediamente $2\log_2(i)$ moltiplicazioni (considerando l'uso del *fast exponentiation algorithm* per il calcolo di g^i). La ricerca delle collisioni tra gli elementi della seconda lista e quelli della prima necessita di $n\log_2(n)$ confronti. Quindi l'algoritmo utilizzato impiega complessivamente

$$4n\log_2(N) + n\log_2(n) = n\log_2(nN^4) \text{ passi.}$$

Prendendo $n \approx 3\sqrt{N}$, il tempo di calcolo risulta essere circa $13.5 \cdot \sqrt{N} \cdot \log_2(1.3 \cdot N)$. □

Per applicare la proposizione 2.3.1 ad un esempio, risolviamo il logaritmo discreto

$$2^x = 81 \quad \text{nel gruppo } \mathbb{F}_{491}.$$

Scegliamo casualmente degli esponenti x tra 1 e 490 (490 è l'ordine di 2) e calcoliamo i valori di 2^x e $81 \cdot 2^x$ fino a quando non troviamo una corrispondenza.

x	2^x	$81 \cdot 2^x$
150	9	**238**
79	443	40
6	64	278
220	260	438
50	289	332
100	51	203
399	347	120
190	97	1
45	454	440
450	**238**	129

Tabella 2.1: Risoluzione di $2^x = 81$ in $\mathbb{F}_{491}$ scegliendo esponenti casuali

Dalla tabella 2.1 è possibile notare

$$2^{450} = 81 \cdot 2^{150} = 238 \quad \text{in } \mathbb{F}_{491}.$$

A questo punto abbiamo trovato la soluzione

$$2^{450} \cdot 2^{-150} = 2^{300} = 81 \quad \text{in } \mathbb{F}_{491}.$$

Capitolo 3

Teoria dell'informazione

Negli anni 1948 e 1949, Claude Shannon pubblicò due articoli, "*A mathematical theory of communication*"[3] e "*Communication theory of secrecy systems*"[4], che ancora oggi sono considerati come le basi matematiche su cui è fondata la crittografia moderna.
In particolare, in [4] Shannon elabora una teoria della sicurezza per i sistemi crittografici assumendo l'assenza di limiti sulle risorse computazionali che un individuo può impiegare per attaccarli.
Shannon dimostra che un sistema crittografico perfettamente sicuro richiede che il numero di chiavi sia almeno pari a quello dei possibili testi in chiaro e che ogni chiave venga utilizzata con uguale probabilità. Questa condizione implica che la maggior parte dei sistemi crittografici pratici non gode di sicurezza incondizionata.
In [3] invece, Shannon elabora una teoria matematica atta a quantificare l'informazione trasmessa da una variabile aleatoria. Qualora tale variabile dovesse rappresentare i possibili testi in chiaro, testi cifrati o chiavi di un cifrario usato per la codifica di un linguaggio naturale, si ottiene una struttura rigorosa per l'analisi matematica della sicurezza crittografica. Shannon definisce tale misura attraverso il concetto di entropia, ispirandosi all'analogia formale con l'entropia di Boltzmann[1] in meccanica statistica e alla concezione del linguaggio come un processo stocastico, ossia un sistema in cui la generazione di sequenze simboliche è regolata dalla probabilità.

3.1 Cifrari simmetrici

Un cifrario simmetrico utilizza una chiave k scelta da uno spazio di possibili chiavi K per cifrare un messaggio m da uno spazio di messaggi M; il risultato del processo di cifratura è un testo cifrato c appartenente ad uno spazio di testi cifrati C.

[1] Ludwig Eduard Boltzmann (1844-1906) è stato un fisico e matematico, le cui idee hanno profondamente influenzato la cultura scientifica del Novecento.

La cifratura può essere vista come una funzione

$$e : K \times M \to C$$

il cui dominio $K \times M$ è l'insieme delle coppie (k, m) mentre il codominio è lo spazio dei testi cifrati C. Analogamente, la decifrazione è una funzione

$$d : K \times C \to M.$$

Spesso è conveniente esprimere la dipendenza di queste funzioni dalla chiave k mediante un pedice. Quindi, per ogni chiave k, le due funzioni possono essere scritte nel seguente modo

$$e_k : M \to C \qquad \text{e} \qquad d_k : C \to M.$$

Inoltre per assicurare che d_k annulli l'effetto di e_k, le due funzioni devono soddisfare la formula

$$d_k(e_k(m)) = m \quad \forall m \in M. \tag{3.1}$$

Questo significa che d_k è l'inversa sinistra della funzione e_k per ogni k. In particolare, la funzione e_k deve essere iniettiva, poiché se $e_k(m) = e_k(m')$:

$$m = d_k(e_k(m)) = d_k(e_k(m')) = m'.$$

3.2 Segretezza perfetta

Un sistema crittografico ha una segretezza perfetta se l'intercettazione di un testo cifrato non fornisce al crittoanalista alcuna informazione né sul testo in chiaro sottostante né su eventuali messaggi cifrati futuri. Introduciamo le variabili aleatorie M, C, K, dove M è una variabile i cui valori corrispondono ai possibili messaggi, C è una variabile i cui valori corrispondono ai possibili testi cifrati e K è una variabile i cui valori sono utilizzate per la cifratura e la decifrazione, con M e K indipendenti.
Siano f_M, f_C, f_K le funzioni di densità di probabilità associate. Queste funzioni sono collegate tra loro attraverso la formula di criptazione/decriptazione (3.1).
Consideriamo inoltre le probabilità congiunte e condizionate di tutte le coppie di queste tre variabili, come $f_{(C,M)}(c, m)$ e $f_{(C|M)}(c|m)$.

Definizione. *Un sistema crittografico ha segretezza perfetta se*

$$f_{(M|C)}(m|c) = f_M(m) \qquad \forall m \in M \ e \ \forall c \in C. \tag{3.2}$$

Questo significa che la probabilità di un determinato messaggio m, $\Pr(M = m)$, è indipendente dal testo cifrato osservato, quindi quest'ultimo non fornisce alcuna informazione sul testo in chiaro. Utilizzando la formula di Bayes la quale afferma che

$$f_{(M|C)}(m|c)f_C(c) = f_{(C|M)}(c|m)f_M(m) \tag{3.3}$$

si può notare che la segretezza perfetta è equivalente alla condizione

$$f_{(C|M)}(c|m) = f_C(c) \qquad \forall c \in C \text{ e } \forall m \in M \text{ con } f(m) \neq 0. \tag{3.4}$$

Da 3.4 segue che la comparsa di un determinato testo cifrato è altrettanto probabile, indipendentemente dal testo in chiaro.
Per una chiave k, la probabilità che il testo cifrato sia c è uguale alla probabilità che la decifrazione di c sia il testo in chiaro, supponendo che c sia il risultato della cifratura di un messaggio con la chiave k.
A questo punto è possibile calcolare $f_C(c)$ sommando su tutte le possibili chiavi e utilizzando la formula di decomposizione

$$\Pr(E) = \Pr(E|F)\Pr(F) + \Pr(E|F^C)\Pr(F^C) \tag{3.5}$$

dove E ed F sono eventi e F^C indica l'evento complementare di F.

Proposizione 3.2.1. *Siano $\mathcal{M}, \mathcal{C}, \mathcal{K}$ rispettivamente l'insieme dei possibili messaggi, l'insieme dei possibili testi cifrati e l'insieme delle possibili chiavi e siano M, C, K delle variabili aleatorie rispettivamente su $\mathcal{M}, \mathcal{C}, \mathcal{K}$ con densità di probabilità associate f_M, f_C, f_K.*
Sia $e_k : \mathcal{M} \to \mathcal{C}$ una funzione di codifica iniettiva per ogni $k \in \mathcal{K}$, e $d_k : \mathcal{C} \to \mathcal{M}$ la corrispondente funzione di decodifica (come descritto nella Sezione 3.1).
Allora, se M e K sono indipendenti, la densità di probabilità di C è data da

$$f_C(c) = \sum_{\substack{k \in \mathcal{K} \\ \text{tale che } c = e_k(m) \\ \text{per qualche } m \in \mathcal{M}}} f_K(k) f_M(d_k(m)) \tag{3.6}$$

Dimostrazione. Per definizione, la densità di probabilità di C in c è

$$f_C(c) = \Pr(C = c) = \Pr(e_k(M) = c).$$

L'evento $e_k(M) = c$ si verifica se e solo se esistono $k \in \mathcal{K}$ e $m \in \mathcal{M}$ tali che

$$K = k \quad \text{e} \quad M = m \quad \text{con} \quad e_k(m) = c.$$

È possibile quindi riscrivere la probabilità come somma sulle coppie (k, m)

$$\Pr(e_k(M) = c) = \sum_{k \in \mathcal{K}} \sum_{\substack{m \in \mathcal{M} \\ e_k(m) = c}} \Pr(K = k,\ M = m).$$

Se per ogni k la funzione e_k è iniettiva allora esiste un unico $m = d_k(c)$ per cui $e_k(d_k(c)) = c$, quindi la somma su m che soddisfa $e_k(m) = c$ può essere ridotta a

$$\sum_{\substack{m \in \mathcal{M} \\ e_k(m) = c}} \Pr(K = k,\ M = m) = \Pr(K = k,\ M = d_k(c)).$$

Quindi,

$$\Pr(e_k(M) = c) = \sum_{\substack{k \in \mathcal{K} \\ e_k(d_k(c)) = c}} \Pr(K = k,\ M = d_k(c)).$$

Essendo M e K indipendenti, allora

$$\Pr(K = k,\ M = d_k(c)) = \Pr(K = k) \Pr(M = d_k(c)) = f_K(k) f_M(d_k(c))$$

Da cui

$$f_C(c) = \sum_{\substack{k \in \mathcal{K} \\ e_k(d_k(c)) = c}} f_K(k) f_M(d_k(c)).$$

Ma, essendo $e_k(d_k(c)) = c$ equivalente alla condizione $e_k(m) = c$ per un qualche $m \in \mathcal{M}$, è possibile riscrivere la formula come segue

$$f_C(c) = \sum_{\substack{k \in \mathcal{K} \\ \text{tale che } c = e_k(m) \\ \text{per qualche } m \in \mathcal{M}}} f_K(k) f_M(d_k(c)).$$

□

Proposizione 3.2.2. *Se un sistema crittografico ha segretezza perfetta, allora* $\#\mathcal{K} \geq \#\mathcal{C}^+$, *dove* $C^+ = \{m \in \mathcal{M} : f_M(m) > 0\}$ *è l'insieme dei messaggi che hanno una probabilità positiva di essere selezionati.*

Dimostrazione. Fissiamo un testo cifrato $c \in \mathcal{C}$ tale che $f_C(c) > 0$. La segretezza perfetta (3.4) dice che

$$f_{(C|M)}(c|m) = f_C(c) > 0 \qquad \forall\, m \in \mathcal{C}^+.$$

Da cui segue che esiste una probabilità positiva che un messaggio $m \in \mathcal{C}^+$ venga cifrato in c; in particolare, esiste almeno una chiave k tale che $e_k(m) = c$.
Inoltre, prendendo un messaggio diverso $m' \in \mathcal{C}^+$ si ottiene una chiave diversa k', poiché altrimenti si avrebbe $e_k(m) = c = e_k(m')$ che va in contrasto con l'iniettività di e_k.
Quindi, ogni messaggio $m \in \mathcal{C}^+$ è associato ad una o più chiavi e messaggi diversi sono associati a chiavi diverse, il che mostra che il numero di chiavi è maggiore o uguale al numero di messaggi in $\mathcal{C}^+$. □

Una scelta particolarmente efficiente, che rispetta le condizioni necessarie, è quella di assumere che gli spazi delle chiavi, dei messaggi, e dei testi cifrati abbiano tutti la stessa cardinalità

$$\#\mathcal{K} = \#\mathcal{M} = \#\mathcal{C}.$$

Sotto tale ipotesi, Shannon dimostra il seguente teorema che caratterizza la segretezza perfetta: di fatto i sistemi crittografici perfettamente segreti non sono pratici.

Teorema 3.2.1. *Supponiamo che un sistema crittografico soddisfi*

$$\#\mathcal{K} = \#\mathcal{M} = \#\mathcal{C}.$$

Allora il sistema ha segretezza perfetta se e solo se valgono le seguenti condizioni:

a) Ogni chiave $k \in \mathcal{K}$ viene usata con uguale probabilità.

b) Per un dato messaggio $m \in \mathcal{M}$ e un dato testo cifrato $c \in \mathcal{C}$ esiste una ed una sola chiave $k \in \mathcal{K}$ che cifra m in c.

Dimostrazione. Supponiamo che il sistema crittografico abbia segretezza perfetta. Verifichiamo la condizione b). Per ogni $m \in \mathcal{M}$ e $c \in \mathcal{C}$, consideriamo l'insieme (eventualmente vuoto) delle chiavi che cifrano m in c,

$$\mathcal{S}_{m,c} = \{k \in \mathcal{K} \ : \ e_k(m) = c\}.$$

Dimostreremo in tre passi che, se il sistema ha segretezza perfetta, allora $\#\mathcal{S}_{m,c} = 1$ per ogni $m \in \mathcal{M}$ e ogni $c \in \mathcal{C}$.
Passo 1. Se $m \neq m'$, allora $\mathcal{S}_{m,c} \cap \mathcal{S}_{m',c} = \emptyset$

Supponiamo che $k \in \mathcal{S}_{m,c} \cap \mathcal{S}_{m',c}$. Allora $e_k(m) = c = e_k(m')$, il che implica che $m = m'$ data l'iniettività della funzione e_k.

Passo 2. Se il sistema crittografico ha segretezza perfetta, allora $\mathcal{S}_{m,c}$ è non vuoto per ogni m e c.

Utilizziamo l'ipotesi di segretezza perfetta nella forma $f_{(M|C)}(m,c) = f_M(m) f_C(c)$. Sappiamo che $f_M(m) > 0$ dato che $m \in \mathcal{M}$ è un testo in chiaro valido per almeno una chiave. Analogamente ogni $c \in \mathcal{C}$ appare come cifratura di almeno un messaggio, quindi $f_C(c) > 0$. Perciò, la segretezza perfetta implica che $f_{(M|C)}(m,c) > 0$ per ogni $m \in \mathcal{M}$ e $c \in \mathcal{C}$. Ma ciò equivale a dire che c è un possibile modo di cifrare m, da cui segue che deve esistere almeno una chiave $k \in \mathcal{K}$ tale che $e_k(m) = c$ e quindi $\mathcal{S}_{m,c}$ non è vuoto.

Passo 3. Se il sistema crittografico ha segretezza perfetta, allora $\#\mathcal{S}_{m,c} = 1$.

Fissato un testo cifrato $c \in \mathcal{C}$. Allora

$$\begin{aligned}
\#\mathcal{K} &\geq \#\left(\bigcup_{m\in\mathcal{M}} S_{m,c}\right) && \text{dato che } \mathcal{K} \text{ contiene ogni } S_{m,c},\\
&= \sum_{m\in\mathcal{M}} \#S_{m,c} && \text{poiché i vari } S_{m,c} \text{ sono disgiunti dal Passo 1},\\
&\geq \#\mathcal{M} && \text{poiché } \#S_{m,c} \geq 1 \text{ dal Passo 2},\\
&= \#\mathcal{K} && \text{dall'assunzione } \#\mathcal{K} = \#\mathcal{M}.
\end{aligned}$$

Di conseguenza, tutte queste disuguaglianze sono in realtà delle eguaglianze. Da

$$\sum_{m\in\mathcal{M}} \#S_{m,c} = \#\mathcal{M}$$

e $\#S_{m,c} \geq 1$ segue che ogni $\#S_{m,c}$ debba essere uguale ad 1. Questo conclude la dimostrazione della condizione b) del teorema.

Passiamo ora alla condizione a). Consideriamo l'insieme di terne

$$(k,m,c) \in \mathcal{K} \times \mathcal{M} \times \mathcal{C} \qquad \text{tali che } \; e_k(m) = c$$

Si nota che k e m identificano univocamente c, e dalla condizione b) sappiamo che anche m e c determinano un unico k. Inoltre usando l'ipotesi $\#\mathcal{M} = \#\mathcal{C}$ si può concludere che c e k identificano univocamente m. A questo punto, per ogni terna (k,m,c) tale che $e_k(m) = c$,

abbiamo

$$
\begin{aligned}
f_M(m) = f_{(M|C)}(m|c) \quad & \text{per la segretezza perfetta,} \\
= \frac{f_{(M|C)}(m,c)}{f_C(c)} \quad & \text{dalla definizione di probabilità condizionata,} \\
= \frac{f_{(M|K)}(m,k)}{f_C(c)} \quad & \text{dato che ogni coppia tra } k, c, m \text{ determina il terzo elemento della terna,} \\
= \frac{f_M(m) f_K(k)}{f_C(c)} \quad & \text{data l'indipendenza di } M \text{ e } K.
\end{aligned}
$$

Dividendo entrambi i membri per $f_M(m)$ si ottiene

$$f_K(k) = f_C(c) \qquad \forall k \in \mathcal{K} \text{ e } \forall c \in \mathcal{C}.$$

Sommando su tutti i $c \in \mathcal{C}$ e dividendo per $\#\mathcal{C}$

$$f_K(k) = \frac{1}{\#\mathcal{C}} \sum_{c \in \mathcal{C}} f_C(c) = \frac{1}{\#\mathcal{C}}. \tag{3.7}$$

Da cui segue che $f_K(k)$ è costante, indipendentemente dalla scelta di $k \in \mathcal{K}$ che corrisponde a quanto affermato dal punto a). (3.7) ci dice anche che $f_C(c)$ è costante e quindi ogni testo cifrato viene impiegato con la stessa probabilità. □

Esempio 3.2.1. Il cifrario di Vigenère analizzato nel capitolo 1 non è perfettamente segreto perché, da (3.4), è richiesto che il testo cifrato non fornisca alcuna informazione sul testo in chiaro. Come visto nel capitolo dedicato, nel cifrario di Vigenère la chiave solitamente è di lunghezza inferiore rispetto al messaggio da cifrare il che permette di effettuare analisi crittografiche (sezioni 1.3 e 1.4) per recuperare il testo in chiaro. La segretezza perfetta potrebbe essere raggiunta dal cifrario di Vigenère solo sotto l'ipotesi di una chiave casuale di lunghezza pari al messaggio, usata una sola volta e successivamente scartata.

Esempio 3.2.2. Il *one-time pad* di Vernam, brevettato nel 1917, è esempio pratico di crittosistema con segretezza perfetta. Si tratta di un sistema semplice ma molto inefficiente. La chiave $k = k_0 k_1 \dots k_N$ consiste in una stringa binaria utilizzata per cifrare un messaggio $m = m_0 m_1 \dots m_N$ mediante l'operazione XOR($\oplus$) bit a bit. Il testo cifrato $c = c_0 c_1 \dots c_N$ è dato da

$$c_i = k_i \oplus m_i \qquad \text{per } i = 0,1,\dots,N.$$

Ogni chiave viene usata una sola volta con uguale probabilità e successivamente scartata (da qui il nome *one-time pad*). Dato che esiste esattamente una chiave, $k = m \oplus c$, che cifra un messaggio m in un testo cifrato c, il teorema 3.2.1 garantisce che il sistema gode di segretezza

perfetta.
Tuttavia, la trasmissione di N bit di testo richiede di aver già condiviso in precedenza N bit di chiave segreta, rendendo il metodo poco pratico su larga scala.
Un altro problema è che il riutilizzo di una chiave compromette la segretezza, come accaduto durante la Seconda Guerra Mondiale quando l'URSS commise questo errore permettendo così agli Stati Uniti di decifrare alcuni documenti.

3.3 Entropia

Nei sistemi crittografici efficienti non è possibile raggiungere la segretezza perfetta dal momento che una singola chiave deve essere utilizzata per cifrare numerosi testi in chiaro. Questo permette di raccogliere informazioni rilevanti sulla chiave attraverso l'analisi di una raccolta di testi cifrati.
Con lo scopo di studiare questo fenomeno, Shannon introdusse il concetto di entropia che permette di misurare l'incertezza di un sistema.

Definizione. *Sia X una variabile aleatoria che assume valori finiti $x_1, x_2, \ldots, x_n$ e siano $p_1, p_2, \ldots, p_n$ le probabilità associate tali che $p_i = f_X(x_i) = \Pr(X = x_i)$. L'entropia $H(X)$ dipende esclusivamente dalle probabilità p_i degli esiti possibili di X nel seguente modo:*

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i. \tag{3.8}$$

Dato che $H(X)$ dipende solo da $p_1, \ldots, p_n$ indicheremo tale valore anche con $H(p_1, \ldots, p_n)$.

Proposizione 3.3.1. *La funzione H (3.8) definita sulla variabile aleatoria discreta a valori finiti soddisfa alle seguenti proprietà:*
***Proprietà** H_1. La funzione H è continua nelle variabili p_i..*
***Proprietà** H_2. Sia X_n una variabile aleatoria uniformemente distribuita su un insieme $\{x_1, \ldots, x_n\}$, cioè X_n ha n possibili valori ciascuno con probabilità $\frac{1}{n}$. Allora*

$$H(X_{n+1}) \geq H(X_n) \qquad \forall\, n \geq 1$$

Quindi con eventi equiprobabili l'incertezza aumenta quando ci sono più eventi possibili.
***Proprietà** H_3. Considerando l'esito di X come una scelta scomponibile in due scelte successive, allora il valore iniziale di H è una somma pesata dei valori di H relativi alle scelte successive.*

Più precisamente si considerino le variabili aleatorie X, Y, Z_i con $i = 1, \ldots, n$ come segue:

$$\begin{aligned} &X : \Omega \to \{x_{ij} : 1 \leq i \leq n \ \textit{e} \ 1 \leq j \leq m_i\}, \\ &Y : \Omega \to \{1, \ldots, n\}, \\ &Z_i : \Omega \to \{x_{ij} : 1 \leq j \leq m_i\}, \end{aligned}$$

e si supponga che $X = Z_Y$, ovvero $X(\omega) = Z_{Y(\omega)} \ \ \forall \omega \in \Omega$, sicché

$$\forall\, i, j : \qquad \Pr(X = x_{ij}) = \Pr(Y = i \ \textit{e} \ Z_i = x_{ij}).$$

Allora

$$H(X) = H(Y) + \sum_{i=1}^{n} \Pr(Y = i) H(Z_i). \tag{3.9}$$

Dimostrazione. **Proprietà H_1.** Sia $p_i \in [0,1]$, osserviamo che la funzione

$$f(p_i) = p_i \log_2 p_i$$

è continua per $p_i > 0$. Per $p_i \to 0^+$ abbiamo

$$\lim_{p_i \to 0^+} p_i \log_2 p_i = 0.$$

Quindi la funzione $f(p_i)$ è continua su tutto $[0,1]$ e, poiché la somma di funzioni continue è essa stessa continua, H è continua nelle variabili p_i.
Proprietà H_2. Sia $p_i = \Pr(X_n = x_i) = \frac{1}{n}$ per ogni $i = 1, \ldots, n$.
Calcoliamo l'entropia

$$H(X_n) = -\sum_{i=1}^{n} p_i \log_2 p_i = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = -n \cdot \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n.$$

Analogamente, per X_{n+1}

$$H(X_{n+1}) = \log_2(n+1).$$

Poiché $\log_2(n+1) \geq \log_2(n)$ per ogni $n \geq 1$, l'entropia aumenta all'aumentare del numero di eventi equiprobabili.
Proprietà H_3. Supponiamo di avere la variabile aleatoria X i cui esiti possono essere scomposti in due scelte successive.

Siano, per ogni $i \in \{1, \ldots, n\}$ e $j \in \{1, \ldots, m_i\}$:

$$p_{ij} = \Pr(X = x_{ij}),$$
$$p_i = \Pr(Y = i) = \sum_{j=1}^{m_i} p_{ij},$$
$$p(j|i) = \frac{p_{ij}}{p_i} \text{ la probabilità condizionata di ottenere } x_{ij} \text{ dato } Z_i.$$

L'entropia di X è data da

$$H(X) = -\sum_{i=1}^{n}\sum_{j=1}^{m_i} p_{ij} \log_2 p_{ij}$$

Notiamo che $p_{ij} = p_i \cdot p(j|i)$. Quindi

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n}\sum_{j=1}^{m_i} p_i p(j|i) \log_2(p_i p(j|i)) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{m_i} p_i p(j|i)[\log_2 p_i + \log_2 p(j|i))] \\ &= -\sum_{i=1}^{n} p_i \log_2 p_i \sum_{j=1}^{m_i} p(j|i) - \sum_{i=1}^{n} p_i \sum_{j=1}^{m_i} p(j|i) \log_2 p(j|i) \\ &= -\sum_{i=1}^{n} p_i \log_2 p_i + \sum_{i=1}^{n} p_i H(Z_i) \end{aligned}$$

dove abbiamo riconosciuto che, per ogni $i \in \{1, \ldots, n\}$,

$$H(Z_i) = -\sum_{j=1}^{m_i} p(j|i) \log_2 p(j|i)$$

è l'entropia della variabile Z_i. Dato che

$$H(Y) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

otteniamo la formula

$$H(X) = H(Y) + \sum_{i=1}^{n} \Pr(Y = i) H(Z_i).$$

□

Esempio 3.3.1. Sia X una variabile aleatoria tale che $X = Z_Y$, dove

- Y è una variabile aleatoria con valori {1,2} e $f_Y(1) = \frac{1}{2},\ f_Y(2) = \frac{1}{2}$

- Z_1 assume il valore X_1 con probabilità 1, Z_2 assume i valori X_2, X_3 con probabilità rispettive $\frac{2}{3}$, $\frac{1}{3}$.

Con $(Y,\, Z_i)$ indipendenti per ogni i.
Calcoliamo l'entropia di X in due modi diversi.

1. Tramite la densità discreta di X:
   X assume i valori x_1, x_2, x_3 e si ha

$$p_1 = f_X(x_1) = \Pr(Y = 1 \text{ e } Z_1 = x_1) = \Pr(Z = x_1|Y = 1)\Pr(Y = 1) = 1 \times \frac{1}{2} = \frac{1}{2}$$

$$p_2 = f_X(x_2) = \Pr(Y = 2 \text{ e } Z_2 = x_2) = \Pr(Z = x_2|Y = 2)\Pr(Y = 2) = \frac{2}{3} \times \frac{1}{2} = \frac{1}{3}$$

$$p_3 = f_X(x_3) = \Pr(Y = 3 \text{ e } Z_3 = x_3) = \Pr(Z = x_3|Y = 3)\Pr(Y = 3) = \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$$

   (si veda la Figura 3.1)
   Si ha quindi

$$\begin{aligned} H(X) &= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{6} \\ &= \frac{1}{2} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{2} - \frac{1}{6}\log_2\frac{1}{3} \\ &= \frac{1}{2} + \frac{1}{6} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{3} \\ &= \frac{2}{3} - \frac{1}{2}\log_2\frac{1}{3} \end{aligned}$$

2. Utilizzando H_3 si ha

$$H(X) = H(Y) + \sum_{i=1}^{2} \Pr(Y = i)H(Z_i).$$

   Ora $H(Y) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = -\log_2\frac{1}{2} = 1$
   $H(Z_1) = 0; \quad H(Z_2) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}$ da cui

$$\begin{aligned} H(X) &= 1 + \frac{1}{2}\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) = 1 - \frac{1}{3}\log_2\frac{2}{3} - \frac{1}{6}\log_2\frac{1}{3} \\ &= 1 - \frac{1}{3}\log_2 2 - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{3} = \frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{3} \\ &= \frac{2}{3} - \frac{1}{2}\log_2\frac{1}{3} \end{aligned}$$

   si ritrova come previsto il risultato ottenuto direttamente.

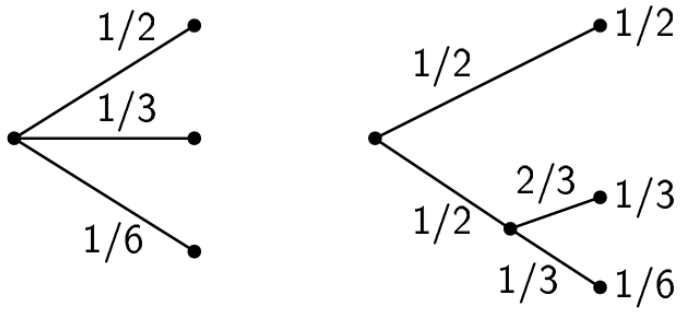

Figura 3.1: Scomposizione di una scelta tra tre possibilità (Esempio 3.3.1).

Teorema 3.3.1. *Ogni funzione che soddisfa le proprietà H_1, H_2, H_3 è necessariamente nella forma:*

$$H = -K \sum_{i=1}^{n} p_i \log_2 p_i, \tag{3.10}$$

con K costante positiva.

Dimostrazione. Sia X_n una variabile aleatoria. Consideriamo inizialmente il caso in cui X_n assume n valori equiprobabili con probabilità $p_i = \frac{1}{n}$ per $i = 1, \ldots, n$ e denotiamo

$$f(n) = H\left(X_n\right).$$

Per la proprietà H_2, $f(n)$ è una funzione monotona crescente in n. Utilizzando la proprietà H_3 possiamo scomporre l'esito di X_n in due scelte successive, la prima con m eventi equiprobabili e la seconda, successiva alla prima scelta, con n eventi equiprobabili. In questo modo otteniamo complessivamente $m \cdot n$ eventi e otteniamo

$$f(m \cdot n) = f(m) + f(n)$$

la cui soluzione, considerate le ipotesi di continuità (H_1) e di funzione monotona (H_2), può essere espressa nella forma

$$f(n) = K \log_2 n \qquad \text{con } K > 0.$$

Quindi nel caso di una distribuzione uniforme X_n abbiamo

$$H\left(X_n\right) = K \log_2 n.$$

Passiamo ora al caso in cui X assuma n valori con probabilità generiche $p_1, \ldots, p_n$.
Supponiamo $p_i = \frac{n_i}{N}$ per ogni $i = 1, \ldots, n$, con $n_i \in \mathbb{N}$ e $\sum_{i=1}^{n} n_i = N$. Applichiamo H_3 per

scomporre l'esito di X_n in due scelte successive: la scelta (Y) di un indice i con probabilità p_i e successivamente, dato i, la scelta (Z_i) di uno tra n_i eventi equiprobabili.

$$H(X) = H(Y) + \sum_{i=1}^{n} p_i H(Z_i)$$

In particolare, indicando al solito con X_k una variabile aleatoria uniforme discreta con k valori,

$$H(X) = H(X_N) + \sum_{i=1}^{n} p_i H(X_n).$$

Utilizzando quanto trovato nel caso di una distribuzione uniforme è possibile riscrivere la formula come segue:

$$H(X) = K \log_2 N - \sum_{i=1}^{n} p_i K \log_2 n_i$$

Poiché $p_i = \frac{n_i}{N}$ si ha

$$\log_2 n_i = \log_2(p_i N) = \log_2 p_i + \log_2 N,$$

da cui

$$H(X) = K \log_2 N - K \sum_{i=1}^{n} p_i (\log_2 p_i + \log_2 N).$$

Sapendo che $\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} \frac{n_i}{N} = 1$ si ha

$$H(X) = K \log_2 N - K \log_2 N - K \sum_{i=1}^{n} p_i \log_2 p_i.$$

A questo punto si conclude che

$$H(X) = -K \sum_{i=1}^{n} p_i \log_2 p_i.$$

□

Al variare della variabile aleatoria con n valori distinti, l'entropia è massima quando la variabile è uniforme.

Corollario 3.3.2. *Sia*

$$f : [0,1]^n \longrightarrow \mathbb{R}$$
$$f(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Sia $T : \{(p_1, \ldots, p_n) \in [0,1]^n : \sum_{i=1}^n p_i = 1\}$.
Allora $\min_T f = 0, \qquad \max_T f = f\left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right) = \log_2 n.$

Dimostrazione. Siccome f è una funzione continua e T è compatto, allora f ha un minimo assoluto e un massimo assoluto per il teorema di Weierstrass.
Chiaramente $f \geq 0$ dove il valore 0 è ottenuto per istanze del tipo $f(\tilde{p},0,0,\ldots,0)$ con $\tilde{p} = 1$.
Sia $\bar{p} = (\bar{p_1}, \ldots, \bar{p_n}) \in T$ tale che

$$f(\bar{p}) = \max_T f.$$

Il teorema di Eulero-Lagrange implica che esiste $\lambda \in \mathbb{R}$ tale che

$$\nabla f(\bar{p}) = \lambda \nabla \left(\sum_{i=1}^n p_i - 1 \right) = \lambda \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \iff \frac{\partial f(\bar{p})}{\partial p_i} = \lambda \qquad \forall\, i \in \{1, \ldots, n\}.$$

Ora,

$$\frac{\partial f(\bar{p})}{\partial p_i} = -\log_2 p_i - 1$$

quindi

$$\log_2 p_i = -(\lambda + 1) \Longrightarrow \log_2 p_i \text{ è costante } \forall\, i \Longrightarrow p_i \text{ è costante } \forall\, i.$$

Dal vincolo $\sum_{i=1}^n p_i = 1$, si deduce che $p_1 = p_2 = \cdots = p_n = \frac{1}{n}$. Quindi il valore massimo di f è raggiunto per

$$f(\bar{p}) = -\sum_{i=1}^n \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n.$$

□

3.4 Ridondanza ed Entropia nel linguaggio naturale

Nel contesto di un testo in chiaro scritto in un linguaggio naturale, come per esempio l'inglese, le lettere adiacenti sono estremamente dipendenti tra loro.
Per mostrare meglio questo concetto, iniziamo stimando l'entropia di una singola lettera in un testo in lingua inglese. Sia L la variabile aleatoria i cui valori corrispondo alle lettere dell'alfabeto inglese E con le probabilità associate riportate nella Tabella 1.6, tra cui:

$$f_L(A) = 0.0817, \quad f_L(B) = 0.0149, \quad \ldots, \quad f_L(Z) = 0.0007.$$

Utilizzando queste probabilità, l'entropia di L è calcolata come:

$$H(L) = -(0.0817 \cdot \log_2 0.0817 + \cdots + 0.0007 \cdot \log_2 0.0007) \approx 4.176. \tag{3.11}$$

Considerando il valore dell'entropia nel caso di lettere equiprobabili, $H = \log_2(26) \approx 4.7$, $H(L)$ risulta essere inferiore in quanto, nella lingua inglese, alcune lettere sono impiegate con maggiore frequenza rispetto ad altre.
Come osservato da Shannon [3], $H(L)$ può essere interpretata come il numero medio di bit di informazione trasmessi da una singola lettera di un linguaggio. Applicando questa osservazione al valore trovato in (3.11) è evidente la presenza di ridondanza data dal fatto che, per rappresentare una lettera dell'alfabeto inglese, sono necessari in media 4.7 bit ma quella stessa lettera trasmette mediamente solo 4.176 bit di informazione.
È necessario precisare che l'entropia $H(L)$ di una singola lettera non tiene in considerazione la correlazione tra lettere adiacenti e per questo motivo non può fornire una stima accurata della ridondanza nella lingua inglese E.
Per includere le dipendenze tra lettere è necessario considerare blocchi di più lettere adiacenti.

Esempio 3.4.1. Definiamo L^3 come la variabile aleatoria i cui valori sono terne ordinate di lettere consecutive in un testo in lingua inglese. Per trovare le frequenze di tali terne è necessario effettuare un conteggio a partire da un testo di riferimento che nel nostro caso è *1984* di George Orwell[2]. Notiamo che alcune terne sono molto più frequenti di altre, per esempio

$$f_{L^3}(THE) \approx 0.0214, \quad f_{L^3}(ING) \approx 0.0078, \quad f_{L^3}(AND) \approx 0.0063.$$

Calcolando ora il valore di $H(L^3)$ otteniamo:

$$H(L^3) \approx 10.85.$$

L'entropia media per lettera si ottiene dividendo il valore ottenuto per la grandezza dei blocchi considerati, che nel nostro caso è uguale a 3; si trova

$$H = \frac{H(L^3)}{3} \approx 3.62.$$

I valori trovati nell'Esempio 3.4.1 sono stati calcolati con lo script Python in Figura 3.2.

Definizione. *Sia $\mathcal{L}$ un linguaggio, e per ogni $n \geq 1$, sia L^n la variabile aleatoria i cui valori sono stringhe di n caratteri consecutivi di $\mathcal{L}$. L'entropia di $\mathcal{L}$ è definita come*

$$H(\mathcal{L}) = \lim_{n\to\infty} \frac{H(L^n)}{n}, \quad \textit{se tale limite esiste.} \tag{3.12}$$

[2]Pseudonimo dello scrittore inglese Eric Arthur Blair(1903-1950)

```python
import re
import math
import json
from collections import defaultdict
def trigram_probability(text):
    # Rimuovi caratteri non alfabetici e trasforma il testo in minuscolo
    clean_text = re.sub(r'[^a-zA-Z]', '', text).lower()

    trigram_count = defaultdict(int)
    total_trigrams = len(clean_text) - 2

    if total_trigrams <= 0:
        return {}

    # Conta l'occorrenza di ogni trigramma
    for i in range(total_trigrams):
        trigram = clean_text[i:i+3]
        trigram_count[trigram] += 1

    # Calcola la probabilità come frequenza relativa
    trigram_probabilities = {trigram: count / total_trigrams for trigram,
    ↪  count in trigram_count.items()}
    return trigram_probabilities

if __name__ == "__main__":
    # Legge il testo da un file (assicurarsi che "input.txt" esista nella
    ↪  stessa directory)
    with open("input.txt", "r") as file:
        text = file.read()
    probabilities = trigram_probability(text)

    # Calcola l'entropia totale (in bit)
    total_entropy = -sum(probability * math.log(probability, 2) for
    ↪  probability in probabilities.values())

    # Ordina i trigrammi per probabilità decrescente
    sorted_trigrams = dict(sorted(probabilities.items(), key=lambda item:
    ↪  item[1], reverse=True))

    # Esporta in un file JSON
    output_filename = "trigram_probability.json"
    with open(output_filename, "w") as json_file:
        json.dump(sorted_trigrams, json_file, indent=4)

    print(f"Risultati esportati in '{output_filename}'")
    print(f"\nTotal Entropy: {total_entropy:.5f} bits")
```

Figura 3.2: Script Python per calcolare le frequenze delle terne e l'entropia

Sebbene non sia possibile determinare con precisione l'entropia della lingua inglese $\mathcal{E}$, sperimentalmente si può arrivare a concludere:

$$1.0 \leq H(\mathcal{E}) \leq 1.5. \tag{3.13}$$

Definizione. *La ridondanza di $\mathcal{L}$ è*

$$R(\mathcal{L}) = 1 - \frac{H(\mathcal{L})}{H(X_{|L|})} \tag{3.14}$$

dove al solito X_m è una variabile aleatoria uniforme con m valori, ed è quindi $H(X_m) = \log_2 m$.

Proposizione 3.4.1. *Sia $\mathcal{L}$ il linguaggio e al solito, per ogni $n \geq 1$, sia L^n la variabile aleatoria i cui valori sono stringhe di n caratteri consecutivi di $\mathcal{L}$.*
Allora $R(\mathcal{L}) \in [0,1]$ (se esiste).

Dimostrazione.

$$R(\mathcal{L}) = 1 - \frac{H(\mathcal{L})}{H(X_{|L|})}$$

$$H(\mathcal{L}) = \lim_{n\to\infty} \frac{H(L^n)}{n}, \quad \text{stiamo supponendo che tale limite esiste.}$$

Ora,

$$H(L^n) \leq H(X_{|L^n|}) = \log_2(|L|^n) = n \log_2(|L|)$$

da cui

$$\frac{H(L^n)}{n} \leq \frac{n \log_2(|L|)}{n} = \log_2(|L|) = H(X_{|L|})$$

sicché

$$\frac{H(\mathcal{L})}{H(X_{|L|})} \leq 1 \text{ e } R(\mathcal{L}) \geq 0.$$

Inoltre

$$\frac{H(L^n)}{n} \geq 0 \text{ e } H(X_{|L|}) \geq 0 \text{ sicché } R(\mathcal{L}) \leq 1.$$

□

Esempio 3.4.2. Se $\mathcal{E}$ è la lingua inglese si ha

$$0.68 \leq R(\mathcal{E}) \leq 0.79$$

Dimostrazione. Infatti, indicando con E la variabile aleatoria i cui valori sono i caratteri di $\mathcal{E}$,

$$\begin{aligned} R(\mathcal{E}) &= 1 - \frac{H(\mathcal{E})}{H(X_{|E|})} \\ &= 1 - \frac{H(\mathcal{E})}{\log_2 26} \\ &= 1 - \frac{H(\mathcal{E})}{4.7} \end{aligned}$$

ed è, per (3.13),

$$\begin{gathered} 1 - \frac{1.5}{4.7} \leq 1 - \frac{H(\mathcal{E})}{4.7} \leq 1 - \frac{1}{4.7} \\ 0.68 \leq R(\mathcal{E}) \leq 0.79. \end{gathered}$$

□

3.4.1 Interpretazione sulla ridondanza

Proposizione 3.4.2. *Sia $\mathcal{L}$ il linguaggio e, per ogni n, sia L^n la variabile aleatoria i cui valori sono le stringhe di n caratteri consecutivi di $\mathcal{L}$. Allora*

$$R(\mathcal{L}) = r \in [0,1[\iff H(L^n) \sim (1-r)H(X_{|L^n|}) \quad \textit{per } n \to \infty$$

$$R(\mathcal{L}) = 1 \iff \forall\, \epsilon \geq 0\ \exists\, \bar{n} \ \textit{ tale che } H(L^n) \leq \epsilon H(X_{|L^n|}) \quad \textit{per } n \geq \bar{n}.$$

Pertanto più la ridondanza è bassa, più l'entropia delle lunghe stringhe di caratteri è alta: il linguaggio è difficilmente prevedibile.
Dualmente, più la ridondanza è alta, più il linguaggio diventa prevedibile.

Dimostrazione.

$$\begin{aligned} R(\mathcal{L}) = r \in [0,1[&\iff \lim_{n\to\infty} \frac{H(L^n)}{n} = (1-r)H(X_{|L|}) \\ &\iff \frac{H(L^n)}{n} \sim (1-r)H(X_{|L|}) \quad \text{per } n \to \infty \\ &\iff H(L^n) \sim (1-r)n\, H(X_{|L|}) = (1-r)H(X_{|L|^n}) \quad \text{per } n \to \infty \\ R(\mathcal{L}) = 1 &\iff \lim_{n\to\infty} \frac{\frac{H(L^n)}{n}}{H(X_{|L|})} = 0 \iff \forall\, \epsilon \ \ \exists \bar{n} : \quad H(L^n) \leq \epsilon\, H(X_{|L|^n}) \quad \text{per } n \geq \bar{n}. \end{aligned}$$

□

Osservazione. La nozione di ridondanza ha risvolti pratici. Una ridondanza pari a $r \in [0,1[$ significa che si può comprimere un messaggio in modo tale che occupi il $100 \times (1-r)\%$ di bit [capitolo 5.6 di [1]].

Esempio 3.4.3. Sia $\mathcal{E}$ la lingua inglese, E come sopra. Per l'esempio si ha $R(\mathcal{E}) \approx 70\%$. Per la Proposizione 3.4.2 si ha quindi $H(E^n) \sim 0.3\, H(X_{|E|^n})$ per $n \to \infty$. Ne segue che l'entropia di E^n è scarsa per n grande: la lingua inglese è piuttosto prevedibile; un messaggio si può comprimere al 30% dei bit originali.

Conclusioni

Questo elaborato ha offerto uno sguardo sui principi fondamentali della teoria dell'informazione e sulla loro applicazione in ambito crittografico. Analizzando l'evoluzione delle tecniche di cifratura è emerso come matematica e statistica costituiscano la base per sviluppare sistemi di comunicazione sicuri ed efficienti.

In particolare, l'analisi del cifrario di Vigenère ha evidenziato come una tecnica, inizialmente ritenuta "inviolabile", possa essere compromessa dall'analisi delle regolarità statistiche. L'approfondimento degli algoritmi di collisione e degli attacchi Meet-in-the-Middle ha dimostrato che la sicurezza di un sistema non dipende soltanto dalla complessità dell'algoritmo utilizzato, ma anche dalla capacità di individuare e sfruttare eventuali debolezze matematiche. I concetti di entropia, ridondanza e segretezza perfetta, invece sottolineano l'importanza di una solida base teorica per comprendere i limiti e le potenzialità dei sistemi di cifratura. Tali concetti, dovuti a Shannon, oltre ad offrire strumenti per la valutazione dell'efficacia degli algoritmi di cifratura, sono fondamentali per la progettazione e l'ottimizzazione di nuove tecniche sempre più sicure ed efficienti.

Abbiamo dimostrato che un sistema crittografico simmetrico (nei quali la chiave utilizzata per crittografare è la stessa usata per decifrare, necessariamente segreta) risulta essere perfettamente sicuro solo nel caso in cui vengano impiegate chiavi diverse per ogni messaggio da cifrare, il che nella pratica risulta essere di difficile gestione. Per quest'ultima ragione, la crittografia moderna si basa su sistemi crittografici asimmetrici con chiave costituita da una parte pubblica ed una parte segreta che non viene scambiata. La teoria dell'informazione formulata da Shannon, con il suo approccio matematico rigoroso, ha sistematizzato in modo rigoroso la comunicazione e la sicurezza dei dati. Le nozioni di entropia e ridondanza viste qui hanno delle conseguenze sulla possibilità di compressione dei bit di un dato linguaggio: più è alta la ridondanza maggiore è la possibilità di compressione.

Bibliografia

[1] J. Hoffstein, J. Pipher e J. H. Silverman, *An Introduction to Mathematical Cryptography*. Springer New York, NY, 2014, isbn: 9781493917105.

[2] F. W. Kasiski, *Die Geheimschriften und die Dechiffrirkunst*. E. S. Mittler und Sohn, 1863.

[3] C. E. Shannon, «A Mathematical Theory of Communication,» *The Bell System Technical Journal*, 1948.

[4] C. E. Shannon, «Communication theory of secrecy systems,» *The Bell System Technical Journal*, 1949.