# Due cifrari a chiave pubblica: Knapsack e McEliece

A cura di:

Alessia Barbagallo
Lucia Cambise
Daniele Di Federico
Cristina Moretti

# Knapsack

A cura di:

Lucia Cambise
Cristina Moretti

# Sommario (1)

- Richiami di crittografia a chiave pubblica
- Definizione di un problema di knapsack
- Cifrario Knapsack
  - *General knapsack*
  - *Superincreasing knapsack*
  - *Merkle – Hellmann knapsack*

# Sommario (2)

- Sicurezza
- Attacco di Shamir al knapsack di Merkle e Hellmann
- Variante del knapsack : Chor - Rivest

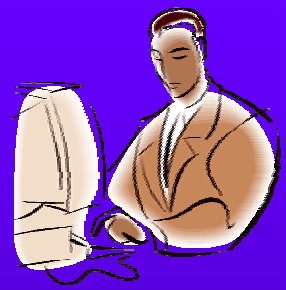
# Richiami: crittografia a chiave pubblica (1)

- Alice vuole spedire un messaggio a Bob:
  - Alice cifra il messaggio in chiaro con la chiave pubblica di Bob e lo invia
  - Bob decifra il messaggio cifrato inviato da Alice con la propria chiave privata
- $K_{pub} \neq K_{priv}$
- Chiunque può cifrare un messaggio per Bob ma solo Bob può decifrarlo

# Richiami: crittografia a chiave pubblica (2)

$K_{pub}$ di Bob

canale di comunicazione



Bob

m → c → m

Alice

$K_{priv}$ di Bob

# Il problema del knapsack: definizioni (1)

1. Siano assegnati un insieme di oggetti $\{1,2,\ldots,n\}$ a ciascuno dei quali è associato un *profitto* $p_j$ ed un peso $w_j$, ed un contenitore *(knapsack)* di *capacità* $W$.

   Il *problema del knapsack* consiste nel selezionare un sottoinsieme di oggetti di profitto massimo da caricare nel contenitore rispettando i limiti imposti dalla capacità $W$.

# Il problema del knapsack: definizioni (2)

… una definizione formale:

$$z^* = \max \sum_{j=1}^{n} p_j x_j$$

$$\sum_{j=1}^{n} w_j x_j \leq W$$

$$0 \leq x_j \leq 1 \text{ intero}, \; j \in \{1,\ldots,n\}$$

# Il problema del knapsack: definizioni (3)

2. Un'istanza del *problema del sottoinsieme somma (subset sum):*

   dati un *insieme* di *numeri interi positivi* $\{a_1, a_2, \ldots, a_n\}$ (chiamato *knapsack set*) e *un intero positivo S* determinare se esiste un sottoinsieme $A_j$ del *knapsack set* la cui somma sia uguale ad *S*.

# Il problema del knapsack: definizioni (4)

… ovvero determinare se esiste

$$x_i \in \{0,1\}, 0 \le i \le n \quad \text{tale che} \sum_{i=1}^{n} a_i x_i = S$$

# Cifrare con un knapsack generico

- Cifra un messaggio con un problema di knapsack
  - Il *knapsack set* è dato da una lista di *numeri interi positivi* $\{m_1,\ldots,m_j\}$
  - Il *plaintext* è una collezione di bit $\{b_1,\ldots,b_n\}$

    quindi ogni carattere è rappresentato secondo la codifica ASCII in binario
  - Il *ciphertext* è la somma $S$

# Come cifrare

- Ipotesi:
  - Alice vuole spedire a Bob il carattere ‘a’ ***(plaintext)***
  - Il ***knapsack set*** è {5, 14, 9, 23, 16, 7, 31, 27}, $j = 8$
  - $n$ deve valere 8, quindi la codifica ASCII di ‘a’ (cioè 97) in binario è 0110 0001

**S = 0*5 + 1*14 + 1*9 + 0*23 + 0*16 + 0*7 + 0*31 + 1*27 = 50**

- Il ***ciphertext*** di ‘a’ è 50

# Come decifrare (1)

- Dato il *ciphertext* 50 e il *knapsack set* (5, 14, 9, 23, 16, 7, 31, 27)
- Proviamo con 50 – 5 = 45
- 45 – 14 = 31 e 31 – 9 = 22, ma non c'è nessun sottoinsieme degli interi rimanenti la cui somma sia 22
- …qualsiasi combinazione di bit che inizi per 1 1 1 è esclusa

# Come decifrare (2)

- Proviamo a porre il primo bit a 0
- Iniziamo con 50 – 14 = 36, 36 – 9 = 27
- 27 è nel knapsack
- Il *plaintext* è 0 1 1 0 0 0 0 1, codifica ASCII per ‘a’ in binario

- Non è sempre così facile!

# Decifrare con un knapsack generico

- Dato il ***ciphertext*** (nel nostro caso 50), il ***plaintext*** può essere ricostruito risolvendo il problema di knapsack:
  - in questo caso non è difficile con una ricerca esaustiva
  - poiché il ***knapsack set*** consiste di solo 8 interi ci sono $2^8 = 256$ combinazioni possibili di 1 e 0
  - e se il knapsack fosse dell' ordine di grandezza di $10^2$ o anche di più?
  - … con una ricerca esaustiva, nel **CASO PEGGIORE,** servirebbero $2^{100}$ tentativi per trovare il ***plaintext***

# Merkle – Hellmann knapsack

- Nato nel 1978
- Algoritmo di cifratura a chiave pubblica
- *Deterministico: fissata una chiave pubblica uno stesso plaintext m viene cifrato in uno stesso ciphertext c*
- Basato sul *superincreasing knapsack*
- NP - completo

# Merkle – Hellmann knapsack: le basi (1)

- Ovviamente il destinatario **legittimo** del messaggio **deve** essere nella condizione di decifrarlo in un tempo accettabile (se non reale)
- L' algoritmo di decifratura è basato su un *problema* di *knapsack* ***facile*** ***("easy/superincreasing knapsack problem")***
- L' algoritmo di cifratura è basato su un *problema* di *knapsack* ***difficile*** ***("hard/non superincreasing knapsack problem")***

# Merkle – Hellmann knapsack: le basi (2)

- La chiave privata è una sequenza di interi positivi di un problema di knapsack facile
- La chiave pubblica è una sequenza di interi positivi di un problema di knapsack difficile (con la stessa soluzione)
- Esiste una tecnica per trasformare un problema di knapsack facile in un problema di knapsack difficile usando un' aritmetica modulare

# Easy/Superincreasing knapsack

- Basato su una *sequenza supercrescente ("superincreasing sequence")* di *interi positivi*
- *Definizione: si dice supercrescente una sequenza* $(b_1, b_2, \ldots, b_n)$ *di interi positivi tale che*

$$b_i > \sum_{j=1}^{i-1} b_j \text{ per ogni } i,\ 2 \le i \le n$$

  - (1, 3, 6, 13, 27, 52) è una sequenza supercrescente
  - (1, 3, 4, 9, 15, 25) non è una sequenza supercrescente

# Come cifrare?

- La cifratura avviene nello stesso modo con cui si cifra con un knapsack generico
- Ad esempio
  - Alice vuole cifrare la lettera ‘R’ (codifica ASCII in binario a 8 bit = 0101 0010) con il knapsack (3, 5, 9, 18, 38, 75, 155, 312)

**S = 0*3 + 1*5 + 0*9 + 1*18 + 0*38 + 0*75 + 1*155 + 0*312 = 178**

- Il *ciphertext* è 178

# Come decifrare?

- INPUT: una sequenza supercrescente $(b_{1,\ldots,} b_n)$ e un intero $S$ che è la somma di un sottoinsieme dei $b_i$
- OUTPUT: $(x_{1,\ldots,} x_n)$ dove $x_i \in \{0,1\}$ t.c. $\sum_{i=1}^{n} x_i b_i = S$

1. $i \leftarrow n$
2. While $i \geq 1$ :
   i. If $S \geq b_i$ then $x_i \leftarrow 1, S \leftarrow S - b_i$
      else $x_i \leftarrow 0$
   ii. $i \leftarrow i - 1$
3. Return $(x_{1,\ldots,} x_n)$

- La soluzione è <u>unica</u> se esiste!

# Un esempio di decifratura

- Dato un knapsack supercrescente (3, 5, 9, 18, 38, 75, 155, 312) e S = 178 decifrare S si traduce nel trovare gli elementi che contribuiscono alla somma

| $x_i$ | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| knapsack | 3 | 5 | 9 | 18 | 38 | 75 | 155 | 312 |
| somme | | | 17 | | | 148 | | |

Codifica ASCII binaria per ‘R’

S = 178

i = 7 knapsack[i] = 155 <= 178

# Da easy a hard knapsack

- Il knapsack supercrescente è facile da decifrare, questo vuol dire che non protegge il messaggio
  - Chiunque può risalire ai bit della sequenza dalla somma S se si conoscono gli elementi della sequenza supercrescente del knapsack
  - Merkle e Hellmann pensarono di convertire l' easy knapsack in un ***trapdoor knapsack*** che non fosse basato su una sequenza supercrescente e quindi fosse difficile da decifrare

# Hard knapsack

- La chiave privata (knapsack set supercrescente) viene utilizzata per generare una chiave pubblica che non è una sequenza supercrescente di interi.
- Con la chiave pubblica si cifra
- Con la chiave privata si decifra

# Generazione della chiave pubblica

- Data la chiave privata $(b_1, \ldots, b_n)$ si sceglie un modulo $M$ t.c. $M > b_1 + b_2 + \ldots + b_n$
- Si sceglie a random $W$, $1 \leq W \leq M - 1$ tale che $\gcd(W, M) = 1$
- Si sceglie una permutazione $\pi$ di interi $\{1,2,\ldots,n\}$ per rendere la sequenza supercrescente una funzione one - way
- Si calcola $a_i = W b_{\pi(i)} \bmod M$, per $i = 1,2,\ldots,n$
- La chiave pubblica è $(a_1, \ldots, a_n)$
- La chiave privata è $(\pi, M, W, (b_1, \ldots, b_n))$

# Riassumendo (1)

- Ogni utente ha una propria *chiave privata* e da questa si ricava la *chiave pubblica*
- Se Alice vuole spedire un messaggio $m$ a Bob deve:
  1. Ottenere la chiave pubblica $E_u = a_1, a_2, \ldots, a_n$ *di Bob*
  2. Rappresentare $m$ in blocchi di $n$ bit tale che $m = m_1, m_2, \ldots, m_n$
  3. Cifrare $m$ in $c$ tale che $c = a_1 m_1 + a_2 m_2 + \ldots + a_n m_n$
  4. Inviare $c$ a Bob

# Riassumendo (2)

- Per ottenere il messaggio in chiaro $m$ a partire da $c$, Bob deve:
  1. Calcolare $d = W^{-1} c \bmod M$
  2. Risolvere un knapsack supercrescnte, trovando gli interi $r_1, r_2, \ldots, r_n$ $r_i \in \{0,1\}$ t.c. $d = r_1 b_1 + r_2 b_2 + \ldots + r_n b_n$
  3. I bit del messaggio sono $m_i = r_{\pi(i)}$, $i = 1,2,\ldots,n$

# Un esempio completo

- $n = 6$
- $(b_{1,\ldots,} b_n) = (12, 17, 33, 74, 157, 316)$
- Scegliamo $M = 737$ e $W = 635$
- $\pi\ (1, 2, 3, 4, 5, 6) = (3, 6, 1, 2, 5, 4)$
- Trovare la chiave pubblica, cifrare e decifrare il messaggio

# Chiave pubblica

- Si inizia da $i = 1$
- Si calcola $a_1 = 635*33 \bmod 737 = 20955 \bmod 737 \equiv 319 \bmod 737$

…

- Fino ad $i = 6$
- La chiave pubblica è

(319, 196, 250, 477, 200, 559)

# Cifrare…

- Alice vuole spedire a Bob il messaggio *m* = “M” che in codifica ASCII binaria a 6 bit è

  101101

- Si cifra:

1. $E_u(M) = 319*1 + 196*0 + 250*1 + 477*1 + 200*0 + 559*1 = 1605$

# …e …

- Alice invia a Bob

$$c = 1605$$

- Bob tramite la sua chiave privata (12, 17, 33, 74, 157, 316) decifra $c$ come segue…

# …decifrare…

- Calcola $d = W^{-1} c \bmod M$ :

  $635^{-1} * 1605 \bmod 737 = 136$

- e risolve il knapsack supercrescente (12, 17, 33, 74, 157, 316)
- 12 + 17 + 33 + 74 = 136
- quindi

$$136 = 12r_1 + 17r_2 + 33r_3 + 74r_4 + 157r_5 + 316r_6$$

$$136 = 12*1 + 17*1 + 33*1 + 74*1 + 157*0 + 316*0$$

. . .

- Riapplicando la permutazione π si ottiene

$r_3 = m_1 = 1$

$r_6 = m_2 = 0$

$r_1 = m_3 = 1$

$r_2 = m_4 = 1$

$r_5 = m_5 = 0$

$r_4 = m_6 = 1$

→ **1 0 1 1 0 1 = *m***

# NP - completezza

- Il problema *subset sum* è NP – completo (dimostrazione per riduzione da *vertex - cover*)
- Il problema *knapsack* è NP – completo (dimostrazione per riduzione da *subset - sum*)
- Tuttavia è stato forzato!

# ATTACCO DI SHAMIR AL KNAPSACK DI MERKLE HELLMAN

Cristina Moretti

# Attacco di Shamir

- Obiettivo: trovare una *coppia di decifrazione* **(W*, M*)** non necessariamente uguale alla coppia (W, M) usata per cifrare
  - Ciò è possibile perché tutti i cifrari knapsack di base hanno *infinite* coppie per la decifrazione: qualsiasi coppia W*/M* sufficientemente vicina a W/M andrà bene.

# Terminologia

- Siano $\{a_i: 1 \leq i \leq n\}$ i pesi del knapsack
- Sia *d* il fattore di espansione, ovvero la misura di quanto i messaggi cifrati siano più lunghi dei messaggi in chiaro (utile per ricavare il valore di M)

$d = \log_2(nM)/n =$

max # di bit del testo cifrato/ # bit nel blocco plaintext

$2^{(dn)} <= M <= 2^{(dn+1)}$

# Algoritmo di Shamir
# fase1

- Calcolare il modulo M approssimandolo a

  $M' = \max \{a_i\}; \quad 1 \leq i \leq n$

  n = # elementi nel knapsack

- Calcolare il fattore di espansione *d* con

  $d^* = (1/n) \log_2 (n^2 M')$

# Algoritmo di Shamir
## fase2

- Impostare $g = d^{*}+2$ o $g = 5$. Il corretto valore di $g$ corrisponde al più piccolo elemento della sequenza superincrementante del problema di knapsack.
- Risolvere il programma intero I.P.:

$$|x_i - x_1 a_i| \leq B\,, \qquad 2 \leq i \leq g$$

$$1 \leq x_1 \leq B\text{-}1 \qquad (1)$$

$$B = [2^{(-n+g)}\, M']$$

# Algoritmo di Shamir fase2

- Se viene trovata una soluzione $(x_1^{(0)}, \ldots, x_1^{(0)})$ creare due nuovi I.P. in cui il vincolo (1) sia sostituito con

$$1 \leq x_1 \leq x_1^{(0)}$$
$$x_1^{(0)} < x_1 \leq B\text{-}1$$

rispettivamente

# Algoritmo di Shamir fase2

- Risolvere questi due I.P. e per ogni soluzione trovata creare altri due nuovi I.P. continuando a suddividere la regione $x_1$ con i valori $x_1^{(i)}$ trovati.
- Iterare fino a trovare $n \log_2 n$ soluzioni distinte, esaminando al massimo $2n \log_2 n$ I.P., oppure finché non vengono nuove soluzioni

# Algoritmo di Shamir
## fase3

- Per ogni soluzione $(x_1^{(i)},\ldots,\ x_n^{(i)})$ trovata nella fase 2, si esamina il razionale $n^7$:

$$t_j^{(i)} = ( x_1^{(i)} / a_i ) + j ( 1 / n^7\, 2^n\, M' )$$

con $\quad 1 \le j \le n^7$

# Algoritmo di Shamir
## fase3

- Trovare $t_j^{(i)} = W_j^* / M_j^*$ usando l'algoritmo Euclideo
- Verificare se $(W_j^* , M_j^*)$ è una coppia di decifrazione per $\{a_1,\ldots, a_n\}$
- Se lo è l'algoritmo termina con successo, altrimenti ricomincia.

# Correttezza

- Dimostriamo che l'algoritmo di Shamir sia corretto
- Dalle formule precedenti, la congruente decifrazione:

$W\ a_i = a_i' \bmod M$ il che equivale a

$W\ a_i - M\ k_i = a_i'$ con $k_i \geq 0$, intero

dividendo tutto per $M\ a_i$ ottengo

$$W/M - k_i/a_i = a_i'/M\ a_i$$

# Correttezza

- Da qui otteniamo:

$$a_1'/Ma_1 - a_i'/Ma_i = k_i/a_i - k_1/a_1$$

dividendo tutto per $a_1 \cdot a_i$

$$k_i a_1 - k_1 a_i = 1/M\,(a_1' \cdot a_i - a_i' \cdot a_1) \quad (2)$$

# Correttezza

- Dal momento che ogni sequenza superincrementante $\{a_i'\}$ con $\Sigma\, a_i' < M$ $(1 \leq i \leq n)$ ha

$$1 \leq a_i' \leq 2^{(-n+i)}\, M \quad (3)$$

Per la (2) e la (3) e per $1 \leq i \leq g$ si ha:

$$| k_i a_1 - k_1 a_i | <= 2^{(-n+g)}\, M'$$

# Correttezza

- E così i I.P. della fase 2 ha al massimo una soluzione $(x_1,\ldots,x_g) = (k_1,\ldots,k_g)$

# Conclusioni

- In questo modo il cifrario di knapsack viene rotto in un tempo polinomiale:

$$O(n^{(g+10)} L \log L) \quad \text{dove } L = g \log M'$$

# CIFRARIO KNAPSACK DI CHOR – RIVEST

Cristina Moretti

# Introduzione

- Il cifrario di Chor-Rivest resistette agli attacchi per molti anni (più a lungo degli altri sistemi)
- E’ uno schema di crittazione knapsack a chiave pubblica che usa più degli altri aspetti matematici complessi

# Richiami

- Un campo (field) è una struttura algebrica in cui è possibile effettuare le operazioni di addizione, sottrazione, moltiplicazione e divisione (eccetto la divisione per 0) e sono soddisfatte le proprietà associativa, commutativa e distributiva
- Più formalmente: un campo è un anello commutativo (F,+,*) t.c. 0 non è equivalente ad 1 e tutti gli elementi di F (eccetto 0) hanno un inverso moltiplicativo

ad esempio numeri reali R

# Algoritmo per la generazione di una chiave pubblica per crittare 1

- Ogni entità crea una chiave pubblica e una corrispondente chiave privata. Per fare ciò ogni entità deve fare le seguenti cose:
  - Selezionare un campo finito $\mathbf{F}_q$ con $p$ caratteristica, dove $q = p^h$, $p \geq h$, e per il quale possibile calcolare il logaritmo discreto
  - Selezionare un polinomio irriducibile $f(x)$ di grado $h$ su $\mathbf{Z}_p$. Gli elementi di $\mathbf{F}_q$ sono rappresentati come polinomi in $\mathbf{Z}_p[x]$ di grado minore a $h$, in modulo $f(x)$

# Algoritmo per la generazione di una chiave pubblica per crittare 2

- Scegliere un elemento primitivo, $g(x)$ del campo $\mathbf{F}_p$
- Per ogni elemento del campo $i \in \mathbf{Z}_q$, trovare il logaritmo discreto $a_i = log_{g(x)} (x + i)$ dell'elemento del campo $(x + i)$ in base $g(x)$
- Scegliere una permutazione $\pi$ sull'insieme degli interi $\{0,1,2,\ldots,p\text{-}1\}$

# Algoritmo per la generazione di una chiave pubblica per crittare 3

- Scegliere un intero qualsiasi $d$, $0 \leq d \leq p^h - 2$
- Calcolare $c_i = (a_{\pi(i)} + d) \bmod (p^h - 1)$, $0 \leq i \leq p - 1$
- La chiave pubblica di A è:

$$((c_0, c_1, \ldots, c_{p-1}), p, h)$$

- La chiave privata di A è:

$$(f(x), g(x), \pi, d)$$

# Esempio: GENERAZIONE DELLA CHIAVE

- A vuole inviare un messaggio cifrato a B
- A sceglie $p = 7$, $h = 4$
- Il polinomio irriducibile di grado 4 su $\mathbf{Z}_7$ è: $f(x) = x^4 + 3x^3 + 5x^2 + 6x + 2$
- Sceglie l'elemento primitivo $g(x) = 3x^3 + 3x^2 + 6$

# Esempio

- Calcola i seguenti logaritmi discreti:

$$a_0 = log_{g(x)} (x) = 1028$$

$$a_1 = log_{g(x)} (x + 1) = 1935$$

.

.

$$a_6 = log_{g(x)} (x + 6) = 223$$

# Esempio

- Scegliere una permutazione random $\pi$ su {0,1,2,3,4,5,6} definita da:

$\pi(0) = 6$ $\quad$ $\pi(4) = 1$

$\pi(1) = 4$ $\quad$ $\pi(5) = 5$

$\pi(2) = 0$ $\quad$ $\pi(6) = 3$

$\pi(3) = 2$

# Esempio

- Scegliere un numero $d = 1702$
- Calcola:

$c0 = (a6 + d) \bmod 2400 = 1925$

$c1 = (a4 + d) \bmod 2400 = 2081$

.

.

$c6 = (a3 + d) \bmod 2400 = 310$

dove $(p^{h} - 1) = 2400$

# Esempio

- Quindi, la chiave pubblica di A è:

$$((c0,c1,c2,c3,c4,c5,c6),\ p = 7,\ h = 4)$$

- La chiave privata di A è:

$$(f(x),\ g(x),\ \pi,\ d)$$

# Esempio: CRITTAZIONE

- Per cifrare il messaggio $m = 22$ per A, B deve fare le seguenti azioni:
  - Ottiene la chiave pubblica autentica di A
  - Rappresenta $m$ come una stringa binaria di lunghezza 5: $m = 10110$
  - Trasforma $m$ nel vettore binario di lunghezza 7 M = {1,0,1,1,0,0,1}
  - Calcola $c = (c_0 + c_2 + c_3 + c_6) \bmod 2400 = 1521$
  - Invia $c = 1521$ ad A

# Esempio: DECRITTAZIONE

- Per decrittare il testo cifrato $c = 1521$ A deve svolgere le seguenti azioni:
  - Calcola $r = (c - hd) \bmod 2400 = 1913$
  - Calcola $u(x) = g(x)^{1913} \bmod f(x) = x^3 + 3x^2 + 2x + 5$
  - Calcola $s(x) = u(x) + f(x) = x^4 + 4x^3 + x^2 + x$
  - Fattorizza $s(x) = x(x+2)(x+3)(x+6)$ (in questo modo $t_1 = 0$, $t_2 = 2$, $t_3 = 3$, $t_4 = 6$)

# Esempio

- I componenti di M di valore 1 hanno indici:
  $\pi^{-1}(0) = 2$
  $\pi^{-1}(2) = 3$
  $\pi^{-1}(3) = 6$
  $\pi^{-1}(6) = 0$
  quindi M = {1,0,1,1,0,0,1}

- Si trasforma M ad intero $m = 22$ trovando il testo originale.

# Sicurezza

- Quando i parametri del sistema sono scelti con attenzione non esiste un schema di attacco al cifrario di Chor-Rivest
  - In particolare sono raccomandati i valori: $p \approx 200$ , $h \approx 25$

# Vantaggi e Svantaggi

- Vantaggi:
  - l'operazione per ricavare il testo cifrato è molto veloce
- Svantaggi:
  - l'operazione di decifrazione è più lenta
  - la chiave pubblica risulta molto grande, circa $(ph \cdot \lg p)$ bit
    - Ad esempio con i paramentri $p = 197$ , $h = 24$ la chiave pubblica è di circa 36000 bit

Daniele Di Federico

# Teoria dei Codici

- Cenni storici
- Codici correttori di errori
- Codici lineari
- Codifica e decodifica
- Teoria dei codici e crittografia
- McEliece

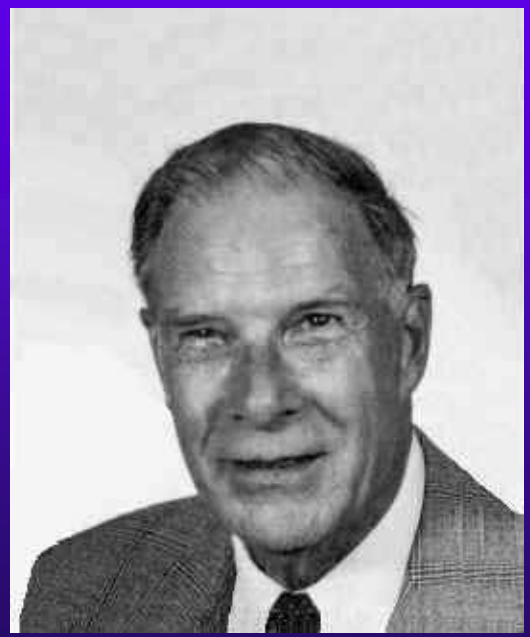
# Cenni storici

" A Mathematical Theory of Comunication" (1948)

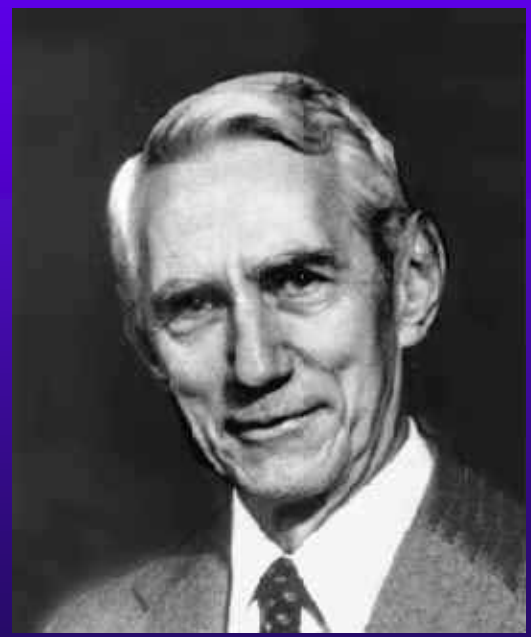
C.E.Shannon

"Error Detecting and Error Correcting Codes" (1950)

R.W.Hamming

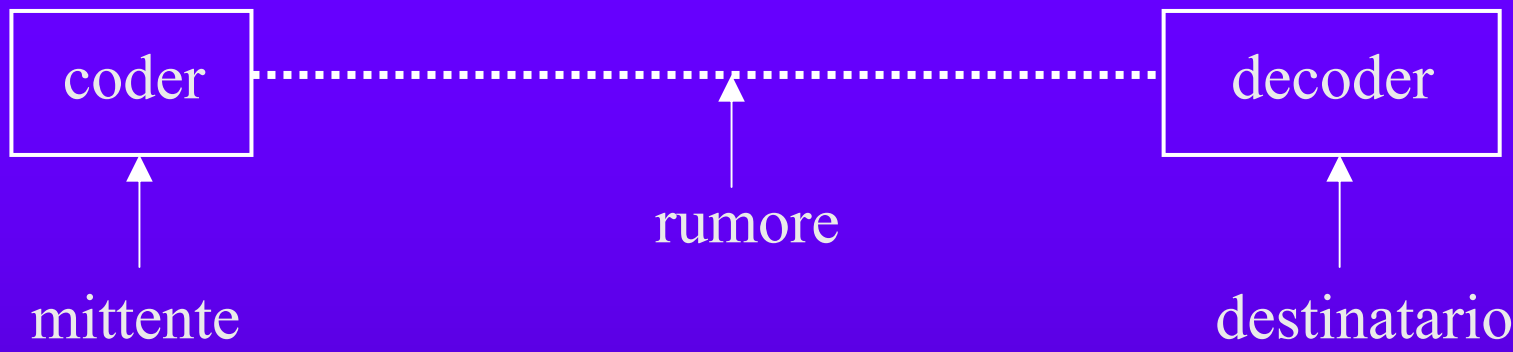


R. W. Hamming



C. E. Shannon

# Comunicazioni in canali rumorosi




L'idea che sta alla base di qualsiasi sistema per la correzione di errori è la ridondanza.

Esempio: sia $p$ la probabilità di avere un errore durante la trasmissione di un bit. Trasmettendo il messaggio due volte otteniamo una probabilità di errore minore, poiché $p<1$ (nel caso binario $p<½$)

# Codici: un po' di teoria…

Def: *Sia $q \in Z$. Si dice "codeword" una sequenza finita di simboli, in cui ogni simbolo appartiene all'insieme $F_q = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$.*

Una codeword puo' essere pensata come un vettore nello spazio $F_q^n = \underbrace{F_q \times F_q \times \ldots \times F_q}_{\text{n volte}}$

Def: *Si dice codice q-ario un insieme di M codeword, dove $M \in N$ è detta dimensione del codice.*

L'insieme di tutte le parole del linguaggio italiano è un codice sull'alfabeto di 21 lettere {a,b,…,z}.

# …ancora teoria…

Def: *La distanza di Hamming d(x,y) tra due codeword x e y di* $F_q^n$ *si definisce come:*

$$d_H(x,y) = |\{i : x_i \neq y_i,\ 0 \leq i \leq n\}|$$

La distanza di Hamming rappresenta il numero di differenze tra una parola di codice e l'altra.

Esempio: x = 0110110 y = 1100101

```
 0110110
 1100101
 ^  ^   ^^
```

*dH(x,y) = 4*

# …sempre teoria…

La distanza di Hamming costituisce una metrica su $F_q^n$ poiché è sempre non negativa e soddisfa:

1. $d(x,y) = 0 \Leftrightarrow x = y$
2. $d(x,y) = d(y,x)$ *per ogni* $x,y \in F_q^n$
3. $d(x,y) \leq d(x,z) + d(z,y)$ *per ogni* $x,y,z \in F_q^n$

Def: *La minima distanza d di un codice è la più piccola distanza di Hamming tra due distinte codeword:*

$$\min\{d(x,y) : x,y \in C,\ x \neq y\}$$

# …ancora sempre teoria…

Con [n,m,d] descriviamo un codice di lunghezza n, dimensione m e avente distanza minima di Hamming d. Tale codice ha un information rate pari a m/n e una distanza minima relativa pari a d/n.

Def: *Un codice C si dice rilevatore di d errori se cambiando al massimo d cifre in una codeword non si produce mai un'altra codeword.*

Def: *Un codice C si dice correttore di e errori se, sapendo che una stringa differisce da qualche codeword di C di al massimo e cifre, è in grado di dedurre la codeword iniziale.*

# Esempio

Repetition binary code: [3,2,3]

C = {(000),(111)}

d((000),(111)) = 3

C è in grado di rilevare 2 errori poiché cambiando 3 cifre ottengo l'altra codeword.

C è in grado di correggere 1 errore. Perché?

Per semplicità ma senza perdere di generalità, in seguito faremo riferimento a codici binari.

# Ancora un po' di teoria

Teorema: *La regola di decodifica sulla massima probabilità afferma che una stringa* $x \in \{0,1\}^n$ *ricevuta da un decoder deve essere decodificata come la codeword con la più piccola distanza di Hamming da x.*

Un codice C con minima distanza di Hamming d puo' essere utilizzato per rilevare fino a d-1 errori oppure a correggere fino a $\lfloor (d-1)/2 \rfloor$ errori

# Esempio

Utilizzando il codice dell'esempio precedente [3,2,3]

C = {(000),(111)}

Utilizzando il codice C come correttore di errori, se arriva il seguente messaggio x = 010, dal teorema della massima probabilità posso decodificare x con 000 poiché:

$$d((000),x) < d((111),x)$$

$$1 < 2$$

# Codici perfetti

Def: *La sfera di Hamming viene definita come:*

$$B(x,r) = \{y : d(x,y) \leq r\}$$

*da cui si osserva che* $|B(x,r)| = |B(\mathbf{0},r)|$ *per ogni x.*

*Una sfera di raggio r in F con* $0 \leq t \leq n$ *contiene esattamente:*

$$V(n,r) = \sum_{k=0}^{r} \binom{n}{k} (q-1)^k \quad \text{vettori}$$

Def: *Hamming bound. Se un codice C è un correttore di e errori, allora vale:*

$$|C| \cdot V(n,e) \leq q^n$$

*Il codice si dice perfetto se vale l'uguaglianza!*

# Codici perfetti (2)

Il limite di Hamming ci dà un limite superiore su come ottenere un buon codice andando praticamente a lavorare sull'information rate. Per ottenere un codice migliore, però, è necessario raggiungere un equilibrio tra information rate e minima distanza relativa.

Def: *(Gilbert, Shannon, Varshamov) GSV bound.*

*Sia A(n,d) la dimensione più grande del codice con minima distanza di Hamming d. Si ha:*

$$A(n,d) \cdot V(n,d-1) \geq q^n$$

# Codici lineari

Def: *Un codice C si dice lineare se per qualsiasi* $u,v \in C$ *allora valgono:*

1. $Au + \beta v \in C$ *per ogni* $\alpha,\beta \in F_q$
2. *Il vettore* **0** *appartiene al codice*
3. *La somma di due codeword qualsiasi in C appartiene ancora a C*

*C rappresenta un sottospazio lineare di* $F_q$

# Anche qui ancora teoria

Def: *Sia C un codice lineare binario* $[n,k,d_{min}]$*, poiché C è un sottospazio vettoriale k-dimensionale avrà una base* $\{\overline{v}_0,\overline{v}_1,\ldots,\overline{v}_{k-1}\}$ *tale che ogni codeword* $\overline{v} \in C$ *possa essere rappresentato come combinazione lineare degli elementi della base:*

$$u_0\overline{v}_0 + u_1\overline{v}_1 + \ldots + u_{k-1}\overline{v}_{k-1}$$

*dove* $u_i \in \{0,1\}$ *e* $0 \leq i < 1$

Questa equazione puo' essere riscritta in altri termini, come vedremo qui di seguito.

# Sempre la solita teoria…

Def: *Data la linearità, possiamo riscrivere l'equazione precedente in termini di una matrice generatrice e un messaggio u come* $\overline{v} = \overline{u} \cdot G$

*con* $G = \begin{pmatrix} \overline{v_0} \\ \overline{v_1} \\ \vdots \\ \overline{v_{k-1}} \end{pmatrix}$

*Poiché C è uno spazio vettoriale k-dimensionale, esiste uno spazio duale* $C^{\perp}$ *(n-k) dimensionale, generato dalle righe di una matrice H tale che* $GH^{\top} = 0$

# …sempre teoria…

*La matrice H è detta matrice di controllo di parità (parity-check matrix) proprio poiché* $GH^{\top} = 0$.

*Si ricava immediatamente che per ogni codeword* $\overline{v}$ *si ha* $\overline{v}\,H^{\top} = \overline{0}$ *(ricordiamo che* $\overline{v} = \overline{u} \cdot G$*)*

*Come vedremo in seguito, questa equazione è di fondamentale importanza per la decodifica di un codice lineare.*

# …ancora teoria…

Una caratteristica negativa dei codici non lineari riguarda il calcolo della distanza minima di Hamming. Infatti è necessario eseguire i confronti con tutte le possibili coppie di codeword $\binom{m}{2}$

Per i codici lineari, si definisce peso di Hamming w(x) la distanza di Hamming dal vettore **0**. Ossia $w(x) = d(x,\mathbf{0})$

Per calcolare il peso di un codice lineare sono necessari solamente m-1 confronti.

# Esempio

*La parity-check matrix puo' sempre essere riscritta nella seguente forma: $G = (I_k|P)$ e di conseguenza si ha $H = (P^T|I_{n-k})$*

Esempio: Si consideri il codice lineare [4,2,2] con matrice generatrice

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Permutando la seconda e quarta colonna otteniamo:

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad \text{ove } P = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{e } H = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

# Encoding

Un codice lineare $C\ [n,k]$ contiene $q^k$ codewords, corrispondenti a $q^k$ messaggi distinti. Identifichiamo con $u = [u_1\ u_2\ \ldots u_k]$ il nostro messaggio, dove $u_i$ sono elementi di $F_q$

Possiamo codificare $u$ moltiplicandolo a destra con la matrice generatrice $G$. Questa operazione mappa $u$ nella combinazione lineare $uG$ delle codewords. In particolare il messaggio con componenti $u_i = \delta_{ik}$ verrà mappato nella codeword alla k-esima riga di $G$.

# Esempio encoding

Esempio: Dato un codice [7,16,3], il messaggio u = [0 1 0 1] e la matrice generatrice G otteniamo:

$$\begin{bmatrix} 0\,1\,0\,1 \end{bmatrix} \begin{pmatrix} 1\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1\,1 \end{pmatrix} = \begin{bmatrix} 0\,1\,0\,1\,1\,0\,0 \end{bmatrix}$$

# Decoding

Due possibili metodi per implementare il modulo per la decodifica dei messaggi codificati:

1. decodifica con array standard (Slepian)
2. Decodifica con sindrome

Vedremo come il primo sia relativamente semplice, ma attuabile solo per codici  lunghezza molto piccola. Il secondo rappresenta il metodo standard per codici con lunghezza n molto grande

# Decoding: array standard

Def: *Sia C un codice lineare su $F_q$ e sia a un qualsiasi vettore di $F_q$.*

*L'insieme $a + C = \{a + x : x \in C\}$ è chiamato coset di C.*

*Proprietà dei coset: (Teorema di Lagrange)*

1. *Ogni vettore di $F_q$ è in qualche coset di C*
2. *Ogni coset contiene esattamente $q^k$ vettori*
3. *Dati due coset qualsiasi, questi sono o equivalenti o disgiunti*

# Slepian

Def: *L'array standard o Slepian di un codice lineare C [n,k] in $F_q$ è un array $q^{n-k}$ x $q^k$ con tutti i coset di C.*

*La prima riga rappresenta proprio le codeword di C. Le righe successive vengono messe una alla volta, iniziando dal vettore con peso minimo in modo progressivo, in modo tale che la entry nella posizione (i,j) è la somma delle entry nelle posizioni (i,1) e (1,j). I vettori della prima colonna sono detti coset leaders.*

# Esempio Slepian

Dato un codice lineare [5,4,3] C con matrice generatrice G:

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

L'array standard Slepian è dato da:

| | | | |
|---|---|---|---|
| 0 0 0 0 0 | 0 1 1 0 1 | 1 0 1 1 0 | 1 1 0 1 1 |
| 0 0 0 0 1 | 0 1 1 0 0 | 1 0 1 1 1 | 1 1 0 1 0 |
| 0 0 0 1 0 | 0 1 1 0 0 | 1 0 1 1 1 | 1 1 0 1 0 |
| 0 0 1 0 0 | 0 1 0 0 1 | 1 0 0 1 0 | 1 1 1 1 1 |
| 0 1 0 0 0 | 0 0 1 0 1 | 0 0 1 1 0 | 0 1 0 1 1 |
| 1 0 0 0 0 | 1 1 1 0 1 | 0 0 1 1 0 | 0 1 0 1 1 |
| 0 0 0 1 1 | 0 1 1 1 0 | 1 0 1 0 1 | 1 1 0 0 0 |
| 0 1 0 1 0 | 0 0 1 1 1 | 1 1 1 0 0 | 1 0 0 0 1 |

# Esempio Slepian (2)

Dato un codice lineare [5,4,3] C con matrice generatrice G:

$$G = \begin{pmatrix} 1\,0\,1\,1\,0 \\ 0\,1\,1\,0\,1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0\,0\,0\,0\,0 \\ 0\,1\,1\,0\,1 \\ 1\,0\,1\,1\,0 \\ 1\,1\,0\,1\,1 \end{pmatrix}$$

L'array standard Slepian è dato da:

| | | | |
|---|---|---|---|
| 0 0 0 0 0 | 0 1 1 0 1 | 1 0 1 1 0 | 1 1 0 1 1 |
| 0 0 0 0 1 | 0 1 1 0 0 | 1 0 1 1 1 | 1 1 0 1 0 |
| 0 0 0 1 0 | 0 1 1 1 1 | 1 0 1 0 0 | 1 1 0 0 1 |
| 0 0 1 0 0 | 0 1 0 0 1 | 1 0 0 1 0 | 1 1 1 1 1 |
| 0 1 0 0 0 | 0 0 1 0 1 | 0 0 1 1 0 | 0 1 0 1 1 |
| 1 0 0 0 0 | 1 1 1 0 1 | 0 0 1 1 0 | 0 1 0 1 1 |
| 0 0 0 1 1 | 0 1 1 1 0 | 1 0 1 0 1 | 1 1 0 0 0 |
| 0 1 0 1 0 | 0 0 1 1 1 | 1 1 1 0 0 | 1 0 0 0 1 |

# Decoding

Poiché il canale di trasmissione puo' introdurre degli errori, la codeword mandata dal coder puo' giungere cambiata al decoder. Se il coder manda la codeword $x$ e il decoder riceve $y$, allora definiamo l'errore $e = y - x$.

Se vi sono stati al massimo t errori, i coset leaders di peso t o minore sono precisamente i vettori errore che possono essere corretti.

# Decoding con array standard

Riprendendo l'esempio precedente, se riceviamo la stringa y = (1 0 1 1 1) e supponiamo che sia stato introdotto al massimo 1 errore, allora …

| | | | |
|---|---|---|---|
| 0 0 0 0 0 | 0 1 1 0 1 | 1 0 1 1 0 | 1 1 0 1 1 |
| 0 0 0 0 1 | 0 1 1 0 0 | 1 0 1 1 1 | 1 1 0 1 0 |
| 0 0 0 1 0 | 0 1 1 1 1 | 1 0 1 0 0 | 1 1 0 0 1 |
| 0 0 1 0 0 | 0 1 0 0 1 | 1 0 0 1 0 | 1 1 1 1 1 |
| 0 1 0 0 0 | 0 0 1 0 1 | 0 0 1 1 0 | 0 1 0 1 1 |
| 1 0 0 0 0 | 1 1 1 0 1 | 0 0 1 1 0 | 0 1 0 1 1 |
| 0 0 0 1 1 | 0 1 1 1 0 | 1 0 1 0 1 | 1 1 0 0 0 |
| 0 1 0 1 0 | 0 0 1 1 1 | 1 1 1 0 0 | 1 0 0 0 1 |

# Decoding con sindrome

Per un codice C di lunghezza n, è necessario effettuare una ricerca su un array di $2^n$ entry per ogni vettore ricevuto. Per codice di lunghezza ragionevole questo sistema è impraticabile.

Introduciamo, dunque, l'altro sistema utilizzato per la decodifica chiamato syndrome decoding descrivendone il funzionamento e le proprietà algebriche che sfrutta.

# Decoding con sindrome (2)

Def: *Sia C un codice lineare [n,k]. Il codice duale* $C^{\perp}$ *di C in* $F_q$ *è l'insieme di tutti i vettori che sono ortogonali ad ogni codeword di C:*

$$C^{\perp} = \{v \in F_q \ : v \cdot u = 0, \text{ per ogni } u \in C\}$$

*Come visto in precedenza,* $H$ *rappresenta la matrice generatrice (n-k)x n per* $C^{\perp}$.

*Il numero* $r = n - k$ *corrisponde al numero di cifre di controllo di parità nel codice, ossia la ridondanza del codice.*

# Decoding con sindrome (3)

Def: *Dato un codice lineare C con parity-check matrix H, il vettore colonna $Hu^T$ è detto sindrome di u*

Def: *Due vettori u e v sono nello stesso coset se e solo se hanno la stessa sindrome.*

C'è una corrispondenza biunivoca tra coset e sindromi!!

# Decoding

Quando un vettore u viene ricevuto, si calcola la sindrome $Hu^T$ e la si confronta con le sindromi dei coset leaders. Se il coset leader con la stessa sindrome è di peso minimo nel coset, allora conosciamo il vettore per decodificare u.

Per calcolare la sindrome per un codice è necessario solamente determinare la matrice H.

Teor: *La sindrome di un vettore che ha un singolo errore m nell'i-esima posizione è m volte la i-esima colonna di H*

# Esempio

Dato il nostro codice [5,4,3], se riceviamo la stringa y = (1 0 1 1 1), allora calcoliamo $Hy^T = (0\ 0\ 1)$

$$con\ H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

*Sempre con l'assunzione che l'errore sia su una cifra, possiamo stabilire che l'errore sia nella quinta cifra e possiamo decodificare y come la codeword (1 0 1 1 0). Esattamente lo stesso risultato lo avevamo ottenuto con l'array standard*

# Teoria dei codici e Crittografia

Se utilizziamo un codice correttore di 1 errore il gioco è molto semplice, abbiamo addirittura un teorema dedicato. Se il codice corregge un numero k di errori con k molto grande, la ricerca della lista di tutte le possibili sindromi diventa onerosa a tal punto da divenire quasi impossibile! (a meno che…)

Crittografia a chiave pubblica: senza la chiave privata e utilizzando dei codici opportuni la decodifica diventa quasi impossibile

# Cenni sui codici di Goppa

In precedenza abbiamo enunciato il teorema di GSV relativo ad un limite che sia migliore di quello di Hamming per i codici.

Per anni non si è mai riusciti a costruire dei codici che fossero in grado di raggiungere il limite imposto dal teorema.

Solamente negli anni ’80, il lavoro di alcuni matematici portò alla scoperta di un particolare tipo di codice i cui parametri erano in grado di “abbattere” il limite.

# Codici di Goppa (2)

Nel 1981 Goppa derivò una classe di codici lineari da curve algebriche su campi finiti che avevano le seguenti proprietà:

1. Sono dei codici abbastanza generali
2. Hanno i parametri circoscritti dal teorema di Riemann-Roch
3. Hanno delle proprietà asintotiche che migliorano il limite classico di GSV
4. L'algoritmo di decodifica è uno tra i più rapidi tra i codici

Barbagallo Alessia

# Sistema di cifratura a chiave pubblica di McEliece

1. Introduzione
2. Sicurezza
3. Algoritmo di generazione delle chiavi
4. Algoritmo di cifratura e decrittazione
5. Esempio
6. Debolezze
7. Attacchi
8. Tuning dei parametri
9. Vantaggi
10. Motivi dello scarso successo
11. Il sistema di cifratura di McEliece oggi
12. conclusioni

# 1. Introduzione

- Proposto nel 1978 da McEliece
- Basato su codici a correzione d'errore
- Idea:
  1) selezionare un particolare codice per il quale è conosciuto un algoritmo di decodifica efficiente
  2) 'mascherare' il codice come un codice lineare
- Una descrizione del codice originale viene utilizzata come chiave privata, mentre una descrizione del codice trasformato serve come chiave pubblica
- Primo schema di cifratura a chiave pubblica ad utilizzare la randomizzazione nel processo di cifratura.

# 2. Sicurezza

- E’ stato mostrato da Berlekamp, McEliece, e van Tilborg che il problema della decodifica di un qualunque codice lineare (trovare una codeword di un dato peso in un codice lineare), è un problema NP-hard (*sindrome del problema della decodifica*)
- Errori introdotti intenzionalmente in un testo cifrato rendono più difficile il lavoro di decrittazione del messaggio

# 3. Algoritmo di generazione delle chiavi (1)

- INPUT: i numeri interi k, n, e t
- OUTPUT: una chiave pubblica e una chiave privata

- Ogni destinatario deve:

a. costruire un codice binario di Goppa C a correzione d'errore facile da risolvere, e scegliere una matrice di generazione G di dimensione k x n per il suddetto codice

b. scegliere a caso una matrice non-singolare e binaria S di dimensione k x k detta `di scramble'.

# 3. Algoritmo di generazione delle chiavi (2)

c. scegliere a caso una matrice P di permutazione di dimensione n x n

d. calcolare la matrice di generazione G ' = SGP di dimensione k x n per un codice lineare binario C' per il quale non è noto alcun algoritmo per la correzione di errori veloce

- La chiave pubblica è G '
- La chiave privata è (S, G, P).

# 4. Algoritmo di cifratura e decrittazione (1)

- Il mittente B cifra un messaggio m destinato ad A il quale svolge la decrittazione

CIFRATURA

B deve eseguire i seguenti passi:

a. ottenere la chiave pubblica G' di A
b. rappresentare il messaggio come una sequenza binaria m di lunghezza k
c. scegliere a caso un vettore di errore binario z di lunghezza n avente al massimo un numero di 1 pari a t
d. calcolare il vettore binario c = mG‘ + z di lunghezza n
e. spedire il messaggio cifrato c ad A.

# 4. Algoritmo di cifratura e decrittazione (2)

## DECRITTAZIONE

Per risalire al testo in chiaro m da c A deve svolgere i seguenti passi:

a. Sapendo che $c = mG' + z = mSGP + z$, calcolare
$c' = cP^{-1} = (mG' + z)P^{-1} = (mSGP + z)P^{-1} =$
$= mSG + zP^{-1}$

b. Usare l'algoritmo di decodifica del codice originale C per decodificare c' ed ottenere
$m' = mS$

c. Calcolare $m = m'S^{-1}$

# 4. Cosa accadrebbe se z fosse nullo?

- Sapendo che $c + z = mG'$, se la matrice G' è invertibile otteniamo $m = (c + z)G'^{-1}$
- Se tutte le componenti del vettore d'errore z fossero nulle avremmo $m = cG'^{-1}$ cosicché il nemico potrebbe recuperare il messaggio in chiaro senza decodifica

# 5. Esempio di cifratura e decrittazione (1)

- Usiamo il codice di Hamming (7,4) a correzione d'errore singolo ($k = 4$, $n = 7$, $t = 1$)
- Una matrice di generazione per questo codice è data da:

$$G = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}$$

# 5. Esempio di cifratura e decrittazione (2)

- Il destinatario sceglie a caso la matrice scramble

$$S = \begin{matrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{matrix}$$

- e la matrice di permutazione

$$P = \begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{matrix}$$

# 5. Esempio di cifratura e decrittazione (3)

- La chiave pubblica del destinatario risulta essere

$$
\begin{array}{rcl}
 & & 1\ 1\ 1\ 1\ 0\ 0\ 0 \\
G' = SGP & = & 1\ 1\ 0\ 0\ 1\ 0\ 0 \\
 & & 1\ 0\ 0\ 1\ 1\ 0\ 1 \\
 & & 0\ 1\ 0\ 1\ 1\ 1\ 0
\end{array}
$$

# 5. Esempio di cifratura e decrittazione (4)

- Un mittente vuole mandare un messaggio

$$m = (1\ 1\ 0\ 1)$$

- prima costruisce un vettore d'errore di peso t

$$z = (0\ 0\ 0\ 0\ 1\ 0\ 0)$$

- calcola il messaggio cifrato y nel modo seguente:

$$c = mG' + z =$$

$$(0\ 1\ 1\ 0\ 0\ 1\ 0) + (0\ 0\ 0\ 0\ 1\ 0\ 0) =$$

$$= (0\ 1\ 1\ 0\ 1\ 1\ 0)$$

# 5. Esempio di cifratura e decrittazione (5)

- il destinatario prima calcola $c' = cP^{-1}$, dove

$$P^{-1} = \begin{matrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

- ottenendo $c' = (1\ 0\ 0\ 0\ 1\ 1\ 1)$

# 5. Esempio di cifratura e decrittazione (6)

- il destinatario decodifica c' mediante l'algoritmo di decodifica.
- Ora il destinatario sa che

$$m' = mS = (1\ 0\ 0\ 0)$$

- può ottenere m moltiplicando per la matrice

$$S^{-1} = \begin{matrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{matrix}$$

- ottenendo $m = mSS^{-1} = (1\ 0\ 0\ 0)S^{-1} = (1\ 1\ 0\ 1)$

# 6. Debolezze (1)

- Berson mostrò che lo schema di McEliece soffre di due debolezze:
  1) fallisce nel proteggere qualunque messaggio che viene cifrato più di una volta,
  2) fallisce nel proteggere dei messaggi che hanno tra loro una relazione lineare nota
- Queste debolezze non rendono immediatamente possibile il forzamento dello schema di McEliece, tuttavia è possibile per un assalitore compiere delle azioni in modo da far sorgere queste debolezze e risalire al messaggio in chiaro

# 6. Debolezze (2)

- Lo schema di cifratura di McEliece può essere indebolito notevolmente se il codice di Goppa viene sostituito con un altro tipo di codice a correzione d'errore
- Gabidulin, Paramonov e Tretjakov mostrarono come sostituendo posto dei codici di Goppa i codici MDR (maximum-rank-distance) lo schema risulti essere poco sicuro

# 7. Attacchi (1)

- Nessuno dei tentativi di forzare lo schema di cifratura di McEliece ha avuto successo
- Nel 1991, Korzhik e Turkin, dichiararono di avere forzato il sistema McEliece mediante un algoritmo che in tempo polinomiale avrebbe risolto il problema della decodifica di un codice lineare
- La descrizione e l'analisi pubblicate non erano precise, inoltre il funzionamento dell'algoritmo stesso entro il tempo dichiarato non è stato mai confermato

# 7. Attacchi (2)

- Due tipi di attacchi che sfruttano due condizioni:
  1) della *rispedizione del messaggio (*va a buon fine in un tempo pari a $\beta k^{3)}$
  2) del *messaggio relazionato* che è una generalizzazione dell'attacco di tipo rispedizione del messaggio (tempo di successo sempre $\beta k^3$)

# 7. Attacchi (3)

## Attacco in condizioni di rispedizione del messaggio

- Condizione di rispedizione del messaggio:

  $c_1 = mSGP + e_1$

  inviati, $e_1 != e_2$

  $c_2 = mSGP + e_2$
- *Profondità* di rispedizione: numero di diversi crittogrammi inviati dello stesso messaggio
- $c_1 + c_2 = e_1 + e_2 \pmod 2$
- Heiman: per scoprire una rispedizione del messaggio basta osservare il peso di Hamming della di qualunque coppia di crittogrammi
  - testi in chiaro diversi ⇨ il peso della somma è circa 512
  - testi in chiaro identici ⇨ il peso della somma non supera 100

# 7. Attacchi (4)
## Metodo d'attacco

- Calcoliamo due insiemi a partire da $c_1 + c_2$:
  - $L_0$ conterrà le posizioni di $c_1 + c_2$ in cui il risultato è uguale a 0
  - $L_1$ conterrà le posizioni di $c_1 + c_2$ in cui il risultato è uguale a 1
- $L_0 = \{l \in \{1, 2, \ldots, 1024\}: c1(l) + c2(l) = e1(l) + e2(l) = 0\}$
- $L_1 = \{l \in \{1, 2, \ldots, 1024\}: c_1(l) + c_2(l) = e_1(l) + e_2(l) = 1\}$

# 7. Attacchi (5)
## Metodo d'attacco

- Si vogliono fruttare le seguenti considerazioni:
  - $l \varepsilon L_0 \Rightarrow$ probabilmente né $c_1(l)$ né $c_2(l)$ sono alterati da un vettore d'errore
  - $l \varepsilon L_1 \Rightarrow$ sicuramente esattamente uno fra $c_1(l)$ e $c_2(l)$ è alterato da un vettore d'errore

# 7. Attacchi (6)
## Metodo d'attacco

- Assumendo che, per ogni l, i vettori d'errore $e_1$ e $e_2$ siano scelti in maniera indipendente

  $\Pr(e_1(l) = 1 = e_2(l) = ( 50/1024 )^2) \sim 0{,}0024$
- In altre parole, la maggior parte degli $l \varepsilon L_0$ indicano che $e_1(l) = 0 = e_2(l)$
- Quindi il crittoanalista deve indovinare 524 colonne non alterate dal vettore d'errore fra quelle contenute in $L_0$

# 7. Attacchi (7)
## Bontà della strategia

- Sia $p_i$ la probabilità che i posizioni siano simultaneamente modificate da $e_1$ e $e_2$. Allora

$$p_i = Pr(|l : e_1(l) = 1 \cap l : e_2(l) = 1| = i) = \frac{\binom{50}{i}\binom{1024-50}{50-i}}{\binom{1024}{50}}$$

# 7. Attacchi (8)
## Bontà della strategia

- Quindi la cardinalità attesa di $L_1$ è

$$E(|L_1|) = \Sigma_{i=0}^{50} (100 - 2i)p_i \sim 95,1$$

visto che ogni l per cui $e_1(l) = 1 = e_2(l)$ riduce $|L_1|$ di due unità

# 7. Attacchi (9)

## Bontà della strategia

- Si supponga che $|L_1| = 94$. Allora $|L_0| = 930$ di cui sono 3 posizioni sono modificate
- La probabilità di indivinare 524 colonne non modificate fra quelle indicate in $L_0$ è

$$\frac{\binom{930-3}{524}}{\binom{930}{524}} \approx 0,0828$$

# 7. Attacchi (10)
## Bontà della strategia

- In questo caso il crittoanalista può aspettarsi di avere successo dopo soli 12 tentativi
- Quando $|L_1| = 96$ sono richiesti solo 5 tentativi

# 7. Attacchi (11)

## Attacco in condizioni di messaggio relazionato

- Condizione di messaggio relazionato:

$c_1 = m_1 SGP + e_1$

inviati, $e_1 != e_2$, $m_1 != m_2$

$c_2 = m_2 SGP + e_2$

- Il crittoanalista può risalire al messaggio $m_i$ dall'insieme $c_i$ svolgendo:
  - una codifica
  - applicando il metodo esposto in precedenza
- Combinando i due messaggi cifrati otteniamo

$$c_1 + c_2 = m_1 SGP + m_2 SGP + e_1 + e_2 = (m_1 + m_2) SGP + e_1 + e_2$$

# 7. Attacchi (12)

## Attacco in condizioni di messaggio relazionato

- A questo punto il crittoanalista risolve

$$c_1 + c_2 + (m_1 + m_2)SGP = e_1 + e_2$$

e procede con il metodo descritto in precedenza usando

$(c_1 + c_2 + (m_1 + m_2)SGP)$ al posto di

$(c_1 + c_2)$

# 8. Tuning dei parametri (1)

- Una scelta accurata dei parametri di sistema k e t può rafforzare l'algoritmo contro gli attacchi aumentando la difficoltà di decodifica
- Nel suo primo articolo, McEliece suggerì n = 1024, t = 50, e k = 524

# 8. Tuning dei parametri (2)

- Molti lavori sono stati svolti allo scopo di studiare i valori ottimali di questi parametri
- La scelta ottima dei parametri del codice di Goppa suggerita prevede a seconda degli autori

$n = 1024$

$524 <= k <= 654$

$37 <= t <= 50$

oppure

$n = 1024$

$k = 644$

$t = 38$

# 8. Tuning dei parametri (3)

- Il nemico, per risalire al messaggio m, deve invertire la matrice di generazione G'
- Assumiamo che tale operazione abbia un costo pari a $k^{\alpha}$, allora il fattore di lavoro conta un numero di passi pari a

$$w = \frac{k^{\alpha}\binom{n}{k}}{\binom{n-t}{k}}$$

# 8. Tuning dei parametri (4)

- Per n = 2i, n, k, e t sono relazionate da
$$k >= 2^i - it = n - it$$
- Quindi per n=1024, abbiamo $k>=1024-10t$
- Si può dimostrare che per α compreso tra 2 e 3, il fattore di lavoro è massimale per t=37
- Per α = 3, il fattore di lavoro vale circa $2^{84,1}$ per t=37, per t= 50 il fattore di lavoro ha un valore pari a $2^{80,7}$
- Prendendo t con il suo valore minimo, si ottiene un incremento del valore minimo di k da 524 a 654

# 9. Vantaggi

- Cifratura e decifratura ad alta velocità
- Cifratura probabilistica (utilizza la regola di decodifica della massima probabilità) unita all'utilizzo dei codici a correzione d'errore rendono il trasferimento dell'informazione accurata e veloce

# 10. Motivi dello scarso successo

- Due motivi principali:
  1) la chiave pubblica ha dimensioni enormi
  2) c'è una espansione del messaggio di un fattore pari a $\begin{pmatrix} n \\ k \end{pmatrix}$
  - Ad esempio, con i parametri di sistema raccomandati n = 1024, t = 38, e k = 644 la chiave pubblica ha una dimensione di circa $2^{19}$ bit e il fattore di espansione del messaggio è pari circa a 1,6

# 11. Il sistema di cifratura di McEliece oggi

- Non si ritiene che i cifrari basati su codici a correzione d'errore possano essere utilizzati per l'autenticazione. A riguardo sono state fatte alcune proposte di firme digitali che utilizzano il sistema di McEliece (es: firma a 111 bit, fattore di lavoro pari a $2^{85}$)
- E' stato proposto un nuovo tipo di attacco, basato su un algoritmo di decodifica statistico, per il quale il sistema di McEliece, utilizzato con i parametri di sistema proposti originariamente, risulta essere non sicuro

# 12. Conclusioni (1)

- La complessità più bassa mai proposta in un attacco ha un numero di passi pari a circa $2^{84}$
- Attacco simile a quello di Shamir applicato su McEliece non porebbe avere successo
- Grazie a cambiamenti nella tecnologia e nell'ambito economico, come ad esempio il crollo dei costi di memorizzazione, si può pensare al sistema di McEliece, come uno dei possibili sistemi di cifratura da utilizzare nelle future applicazioni

# 12. Conclusioni (2)

- L'incremento della potenza di calcolo e di memorizzazione possono essere armi da utilizzare in un attacco contro il sistema stesso
- Soluzione: incrementare ancora i parametri di sistema
- Tuttavia la dimensione della chiave pubblica risulterebbe essere troppo grande per qualunque applicazione pratica