NEWdivcrypto_step1

Benvenuti! Ci incontriamo per parlare di ***crittografia, algoritmi, algebra***. Argomenti affascinanti che, se vorrete, potrete incontrare nel corso di laurea in Matematica.

La crittografia consiste nel trasformare un messaggio in modo che se pure dovesse cadere in "mani nemiche" questo non potrebbe facilmente svelare il suo contenuto.

Immaginate un generale che dal quartier generale invia un messaggio al fronte di guerra. Se fosse inviato "in chiaro" invece che "cifrato" e il nemico fosse in grado di intercettarlo (cattura il messaggero, riceve la comunicazione radio, si inserisce sulla connessione internet, etc) allora questo potrebbe leggerne il contenuto e prendere le contromisure.

Non dobbiamo però solo pensare alle applicazioni militari. Immaginate di essere un investitore che deve inviare in borsa un ordine di acquisto di un grosso quantitativo di azioni. Se gli altri investitori venissero a conoscenza delle sue intenzioni e si convincessere quindi che si tratta di un buon affare allora potrebbero acquistare le azioni prima di lui.

Anche nella vita di tutti giorni noi facciamo uso della crittografia. Avete mai visto una partita di calcio con un decoder satellitare? Se la partita è "a pagamento" allora la trasmissione viene inviata dal

satellite in modo cifrato, criptato così che solo il pubblico pagante possa goderne.

Convinti dell'importanza della crittografia? Se gli Alleati (avere visto il film ***The Imitation Game*** sulla vita del matematico Alan Turing?) non avesse trovato il modo di decifrare i messaggi inviati dai nazisti con la celebre macchina ***Enigma***, la guerra in Europa sarebbe durata molto più a lungo, qualche nostro bisnonno sarebbe stato nei guai e forse noi non saremmo neppure qui. Un buona ragione, secondo me, per capirne qualcosa.

Capire ed agire in modo razionale significa "fare scienza". Il linguaggio della scienza è la matematica. Cerchiamo quindi di spiegare con la matematica cosa è un sistema di crittografia, un cifrario. Partiamo da un esempio, uno dei cifrari più antichi della storia, utilizzato *nientepopodimenoche* da Giulio Cesare (I sec. a.C.) per inviare messaggi alle sue truppe impegnate in Gallia. Per illustrare questo e altri sistemi di crittografia utilizziamo il programma con cui sto lavorando adesso, ***SageMath***, un grosso programma di calcolo simbolico (algebra, geometria, derivate, integrali definiti e indefiniti, etc) completamente gratuito e disponibile su internet

http://www.sagemath.org/

Lo potete scaricare gratuitamente sul vostro

computer (linux, windows, mac os, etc) oppure lavorarci "sulle nuvole" cioè in remoto collegandosi al sito

http://cloud.sagemath.com/

Per semplicare le cose, diciamo che Cesare scriveva in Italiano che consiste di 21 lettere. Formiamo allora una lista (alfabeto) con le lettere dell'Italiano

```
alfa = 'ABCDEFGHILMNOPQRSTUVZ'
```

Scriviamo un procedura, un algoritmo, in questo caso molto semplice, che determina la posizione di una lettera nell'alfabeto. Si tratta un ciclo (while) di ricerca con un contatore (i = i + 1) che aggiorna la posizione corrente. Avete mai programmato? Se non l'avete ancora fatto, fatelo subito! Il mondo digitale intorno a voi è stato creato da programmatori che hanno trasformato funzioni matematiche in procedure. Vorreste non partecipare a tale "creazione"?

```
def position(x,l):
  i = 0
  y = l[i]
  while y <> x:
    i = i + 1
    y = l[i]
  return i
```

```
position('A', alfa); position('B', alfa); position('Z', alfa)
```

```
0
1
```

```
20
```

Possiamo dunque identificare l'alfabeto con l'insieme Z_21 = {0,1,...,20}. Su questo insieme definiamo una somma che è come la somma sull'insieme dei numeri interi Z ma con l'aggiunta della nuova proprietà che "21 = 0". Se ammettiamo questo, allora

$$15 + 10 = 25 = 21 + 4 = 4,$$

$$15 + 15 = 30 = 21 + 9 = 9$$

etc. In altri termini, quando si somma x + y in Z_21 il risultato è il *resto* di x + y in Z rispetto alla *divisione* per 21. Questa idea vale in generale per Z_n = {0,1,...,n-1} dove n è un numero qualsiasi. L'insieme Z_n si chiama **l'insieme dei numeri interi modulo n**. Torniamo a Cesare. Innanzitutto, trasformiamo un messaggio in una lista (insieme ordinato) di interi modulo 21. Costruiamo una procedurina che lo fa per noi.

```
def convert(m):
  l = [position(x,alfa) for x in m];
  return l
```

```
m = 'ATTACCATEVERCINGETORIGE';
l = convert(m);
m; l;
```

```
'ATTACCATEVERCINGETORIGE'
[0, 17, 17, 0, 2, 2, 0, 17, 4, 19, 4, 15, 2, 8, 11, 6, 4, 17,
15, 8, 6, 4]
```

Possiamo facilmente ricovertire la lista di interi nel messaggio originale prendendo dall'alfabeto le lettere che hanno le posizioni contenute nella lista.

In altri termini, abbiamo la procedura (funzione) inversa di conversione

```
def convert_inv(l):
  m = ''
  for x in l:
     m = m + alfa[x]
  return m
```

```
m = convert_inv(l);
l; m;
```

```
[0, 17, 17, 0, 2, 2, 0, 17, 4, 19, 4, 15, 2, 8, 11, 6, 4, 17,
15, 8, 6, 4]
'ATTACCATEVERCINGETORIGE'
```

Questa conversione di lettere in numeri **non è crittografia**. Tutti conoscono le posizioni delle lettere nell'alfabeto, anche i barbari Galli! Vogliamo invece che Cesare e i suoi generali condividano un segreto, una **chiave** che permetta solo a loro di **cifrare (nascondere)** e **decifrare (svelare)** il messaggio. Supponiamo che questa chiave sia una lettera dell'alfabeto, diciamo D. Nell'insieme Z_21 questa lettera vale il numero 3

```
position('D', alfa);
```

```
3
```

Per cifrare possiamo allora sommare 3 a tutti gli interi della lista che corrispondono al messaggio. Per restare in Z_21 la somma deve essere quella di Z_21 cioè dobbiamo ricordare che 21 = 0, 22 = 1, 23 = 2, etc, cioè sostituiamo ad un numero il suo resto nella divisione per 21. In SageMath la procedura che restituisce il resto di x nella divisione per y è x%y

```
21%21; 22%21; 23%21
```

```
0
1
2
```

Quindi la somma in Z_21 è (x+y)%21

```
(15 + 10)%21; (15 + 15)%21
```

```
4
9
```

Cesare somma il segreto 3 alla lista di interi che corrispondono al suo messaggio. Ne ottiene una nuova lista di interi che corrispondo ad un messaggio cifrato che invia ai suoi generali.

```
def add_list(l, y):
  nl = [ (x + y)%21 for x in l]
  return nl
```

```
m = 'ATTACCATEVERCINGETORIGE';
l = convert(m);
m; l;
```

```
'ATTACCATEVERCINGETORIGE'
[0, 17, 17, 0, 2, 2, 0, 17, 4, 19, 4, 15, 2, 8, 11, 6, 4, 17,
15, 8, 6, 4]
```

```
nl = add_list(l,3)
convert_inv(l); convert_inv(nl);
```

```
'ATTACCATEVERCINGETORIGE'
'DZZDFFDZHBHUFNQLHZRUNLH'
```

Ecco, **questa è crittografia**. Solo chi conosce la chiave D = 3 può ottenere il messaggio originale, "in chiaro" sottraendo il numero 3.

```
l = add_list(nl,-3)
convert_inv(l);
```

```
'ATTACCATEVERCINGETORIGE'
```

Componiamo ora tutte queste procedure per

ottenere un singola funzione per cifrare ed una per decifrare secondo il metodo di Cesare (cifrario a scorrimento, shift cipher).

```
def encrypt(m, k):
  x = position(k, alfa)
  l = convert(m)
  nl = add_list(l,x)
  nm = convert_inv(nl)
  return nm
```

```
dm = encrypt(m, 'D'); zm = encrypt(m, 'Z')
m; dm; zm
```

```
'ATTACCATEVERCINGETORIGE'
'DZZDFFDZHBHUFNQLHZRUNLH'
'ZSSZBBZSDUDQBHMFDSNQHFD'
```

La funzione di decifratura si ottiene da quella di cifratura sostituendo alla chiave il suo opposto (in Z_21)

```
def decrypt(m, k):
  x = position(k, alfa)
  y = - x%21
  nk = alfa[y]
  nm = encrypt(m, nk)
  return nm
```

```
m1 = decrypt(dm, 'D'); m2 = decrypt(zm, 'Z')
m1; m2
```

```
'ATTACCATEVERCINGETORIGE'
'ATTACCATEVERCINGETORIGE'
```

Dunque, se Cesare ed i suoi generali condividono una stessa chiave segreta (una lettera) allora sono in grado di scambiarsi messaggi in modo cifrato. Il fatto è che il cifrario di Cesare poteva essere inaccessibile ai barbari del I sec. a.C. ma non certo a noi che viviamo nel XXI sec. d.C. Una grave debolezza di questo sistema è ad esempio il fatto

che il numero di chiavi è molto basso (21 lettere dell'alfabeto) e quindi si può procedere per tentativi provando tutte le chiavi. Supponiamo di catturare un messaggio cifrato.

```
m = 'RUTEFGOPZFTEAZALFFPZFT'; m;
```

```
'RUTEFGOPZFTEAZALFFPZFT'
```

Possiamo eseguire un ciclo "for" per provare a decifrarlo con tutte le chiavi possibili

```
for x in alfa:
    nm = decrypt(m, x)
    print(x, nm)
```

```
('A', 'RUTEFGOPZFTEAZALFFPZFT')
('B', 'QTSDEFNOVESDZVZIEEOVES')
('C', 'PSRCDEMNUDRCVUVHDDNUDR')
('D', 'ORQBCDLMTCQBUTUGCCMTCQ')
('E', 'NQPABCILSBPATSTFBBLSBP')
('F', 'MPOZABHIRAOZSRSEAAIRAO')
('G', 'LONVZAGHQZNVRQRDZZHQZN')
('H', 'INMUVZFGPVMUQPQCVVGPVM')
('I', 'HMLTUVEFOULTPOPBUUFOUL')
('L', 'GLISTUDENTISONOATTENTI')
('M', 'FIHRSTCDMSHRNMNZSSDMSH')
('N', 'EHGQRSBCLRGQMLMVRRCLRG')
('O', 'DGFPQRABIQFPLILUQQBIQF')
('P', 'CFEOPQZAHPEOIHITPPAHPE')
('Q', 'BEDNOPVZGODNHGHSOOZGOD')
('R', 'ADCMNOUVFNCMGFGRNNVFNC')
('S', 'ZCBLMNTUEMBLFEFQMMUEMB')
('T', 'VBAILMSTDLAIEDEPLLTDLA')
('U', 'UAZHILRSCIZHDCDOIISCIZ')
('V', 'TZVGHIQRBHVGCBCNHHRBHV')
('Z', 'SVUFGHPQAGUFBABMGGQAGU')
```

Notate che l'unico messaggio dotato di senso si ottiene in corrispondenza della lettera L. Questo semplice tipo di attacco ad un cifrario (provare tutte le chiavi) constituisce un esempio di quello che si chiama **crittoanalisi**. Questa scienza

consiste nell'indagare la **sicurezza** di un sistema di crittografia determinando la **complessità** di portare un attacco al sistema. La crittoanalisi è generalmente molto complicata e richiede la conoscenza di molta matematica (algebra, teoria dei numeri, probabilità, etc). L'esempio del cifrario di Cesare suggerisce un principio elementare della crittoanalisi:

un cifrario non è sicuro se il numero delle chiavi è basso.

Non è però sempre vero il contrario, come dimostreremo ora.