NEWdivcrypto_step2

Vogliamo fare l'esempio di un altro cifrario elementare ma con un alto numero di chiavi. Ripartiamo daI cifrario di Cesare mostrando che si basava su sostituzioni di lettere in lettere di un tipo speciale. Ridefiniamo nuovamente le funzioni che abbiamo creato per questo cifrario per poterle usare nuovamente.

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

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

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

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

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

Vediamo qual'è l'effetto di applicare il cifrario di Cesare all'alfabeto.

```
alfa = 'ABCDEFGHILMNOPQRSTUVZ'
encrypt(alfa, 'A'); encrypt(alfa, 'B'); encrypt(alfa, 'C');
encrypt(alfa, 'D'); encrypt(alfa, 'E');
```

```
'ABCDEFGHILMNOPQRSTUVZ'
'BCDEFGHILMNOPQRSTUVZA'
'CDEFGHILMNOPQRSTUVZAB'
'DEFGHILMNOPQRSTUVZABC'
'EFGHILMNOPQRSTUVZABCD'
```

Questo ci dice che questo cifrario corresponde a sostituire le lettere di un messaggio con altre lettere dell'alfabeto secondo una corrispondenza uno-ad-uno di tipo particolare. La corrispondenza è ottenuta mandando le lettere dell'alfabeto ordinato in modo naturale (A,B,C,D,...) nelle lettere dell'alfabeto riordinato in modo che la prima lettera sia proprio la chiave e tutte le altre seguano in ordine naturale. Naturalmente quando si arriva a Z si ricomincia con A (cifrario a scorrimento, shift). Questo è esattamente ciò che si ottiene quando si somma un numero di Z_21 a tutti gli altri numeri ordinati in modo naturale (0,1,2,3,...).

```
Z21 = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
add_list(Z21, 0); add_list(Z21, 1); add_list(Z21, 2);
add_list(Z21, 3); add_list(Z21, 4);
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
19, 20]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
20, 0]
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
0, 1]
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
1, 2]
[4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
2, 3]
```

Se la chiave è ad esempio D e quindi la corrispondenza di Cesare è

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

```
'ABCDEFGHILMNOPQRSTUVZ'
'DEFGHILMNOPQRSTUVZABC'
```

otteniamo che la parola GALLIA si trasforma in LDOONL

```
m = 'GALLIA'; m; encrypt(m, 'D')
```

```
'GALLIA'
'LDOOND'
```

Per aumentare il numero di chiavi, potremmo quindi pensare ad un nuovo cifrario basato su **tutte** le possibili sostituzioni di lettere in lettere. Naturalmente queste sostituzioni devono essere corrispondenze uno-ad-uno, **permutazioni**, in modo da permettere la decifratura. Per fare un esempio, le permutazioni dell'alfabeto (ridotto) [A,B,C] sono

[A,B,C]

[B,C,A]

[C,A,B]

[A,C,B]

[C,B,A]

[B,A,C]

Di queste 6 permutazioni, quelle di Cesare (a scorrimento, shift) sono solo le prime 3. In generale, si dimostra che il numero di permutazioni di Z_n è il *fattoriale*

$$n! = n*(n-1)*...*2*1.$$

mentre quelle a scorrimento sono ovviamente solo n. Notiamo che per n = 21 abbiamo

$$21! = 51090942171709440000$$

Il fattoriale è un numero che cresce molto rapidamente, come ben sapevano i costruttori di Enigma che si basava appunto su permutazioni di lettere. Costruiamo ora le procedure necessarie a realizzare il nostro **cifrario a sostituzione**. La chiave è una qualsiasi permutazione dell'alfabeto ovvero di Z_21. Per comodità utilizzerò permutazioni di numeri.

```
def encrypt2(m, k):
  nm = ''
  for x in m:
    i = position(x,alfa)
    j = k[i]
    nm = nm + alfa[j]
  return nm
```

Facciamo una prova con l'alfabeto ridotto.

```
alfa = 'ABC'
m = 'ABBA'; k = [1,0,2]
encrypt2(m, k)
```

```
'BAAB'
```

Per ottenere la decifrazione dobbiamo semplicemente calcolare la permutazione, funzione inversa.

```
def invperm(p):
  np = [position(i,p) for i in range(0,21)]
  return np
```

Facciamo una prova. Prendiamo la permutazione

che sposta i numeri di Z_21 di una posizione in avanti. La permutazione inversa li sposta di una posizione all'indietro.

```
p = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,0];
p; invperm(p)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
20, 0]
[20, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
18, 19]
```

La funzione di decifratura si ottiene allora semplicemente applicando la cifratura mediante la sostituzione inversa.

```
def decrypt2(m, k):
  nk = invperm(k)
  nm = encrypt2(m, nk)
  return nm
```

Adesso ci possiamo divertire. Scegliamo un messaggio e prendiamo una permutazione a caso.

```
alfa = 'ABCDEFGHILMNOPQRSTUVZ'
k = [19,0,5,7,6,12,15,11,4,20,16,2,18,9,1,10,3,8,13,14,17]
m = 'FAREMATEMATICAMIDIVERTEPARECCHIO'; nm = encrypt2(m, k)
len(Set(k)); nm;
```

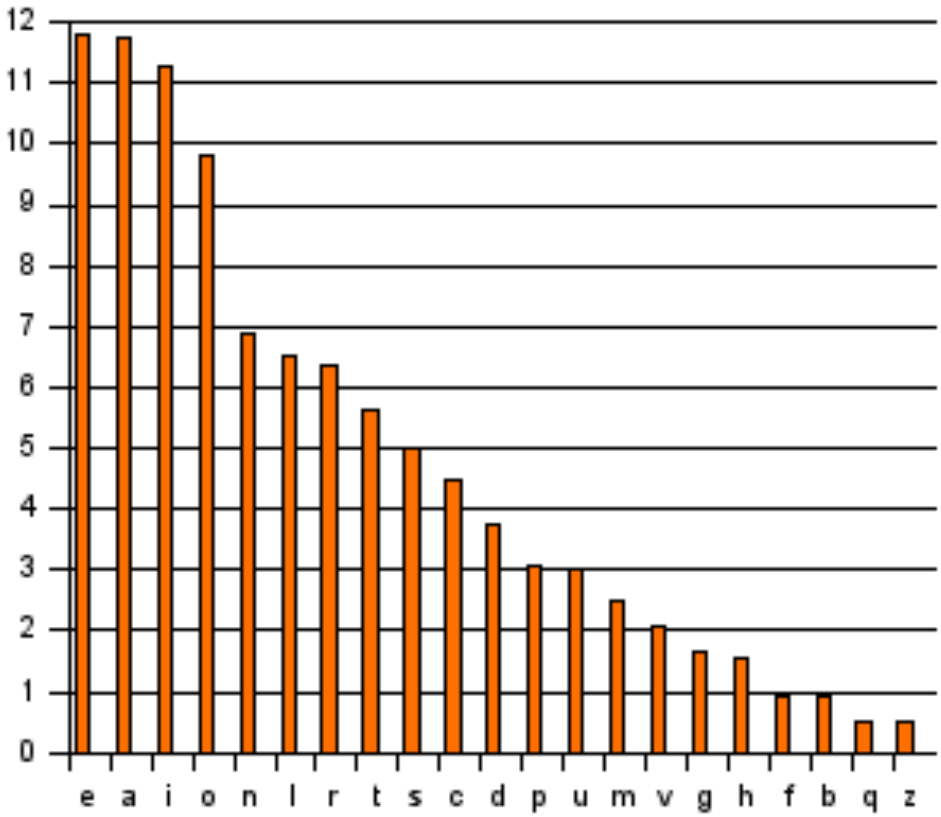
```
21
'OVMGSVIGSVIEFVSEHEQGMIGLVMGFFNEU'
```

```
decrypt2(nm, k)
```

```
'FAREMATEMATICAMIDIVERTEPARECCHIO'
```

Nonostante l'elevato numero di chiavi possibili, neanche un cifrario a sostituzione (da solo) garantisce una sufficiente sicurezza. Enigma, ad esempio, utilizzava un meccanismo che per ogni lettera cifrata modificava la sostituzione (cifrari a flusso). La crittoanalisi ci spiega che il motivo

principale della debolezza di un cifrario a sostituzione è che risulta monoalfabetico cioè lettere in chiaro distinte corrispondono a lettere cifrate distinte (corrispondenza uno-ad-uno). Nei linguaggi naturali come l'Italiano, le singole lettere compaiono nei testi con differenti frequenze. Per un generico testo abbastanza lungo si può calcolare, per esempio in termini di percentuale, la frequenza relativa delle singole lettere dell'alfabeto.



Come si vede, le vocali A, E, I hanno la probabilità (frequenza relativa = frequenza/totale lettere) più elevata di comparire in un testo italiano. Siamo intorno al 12%. La vocale 0 ha probabilità circa 10% e la consonante N circa il 7%.

Un cifrario a sostituzione che è una permutazione

di lettere ***non modifica questa probabilità*** e quindi le lettere che compaiono con frequenza più elevata in un cifrato (abbastanza lungo) di un messaggio in Italiano devono essere necessariamente il cifrato delle lettere A, E, I. Questo ed altre considerazioni simili permettono di abbassare il numero di permutazioni possibili, chiavi ammissibili ad una valore tale che risulta praticabile provare tutte le chiavi (come abbiamo fatto per il cifrario di Cesare). Costruiamo qualche procedura per verificare quanto abbiamo detto.

```
def relfreq(m, x):
  n = len(m)
  i = 0
  for y in m:
    if y == x:
      i = i + 1
  i = (i/n)
  i = i * 100.0
  return i
```

Consideriamo come messaggio le ultime frasi che ho appena scritto e calcoliamo le frequenze relative delle lettere A,E,I,O,N.

```
m =
'UNCIFRARIOASOSTITUZIONECHEEUNAPERMUTAZIONEDILETTERENONMODIFICAQU
m
```

```
'UNCIFRARIOASOSTITUZIONECHEEUNAPERMUTAZIONEDILETTERENONMODIF
APROBABILITAEQUINDILELETTERECHECOMPAIONOCONFREQUENZAPIUELEVA
FRATOABBASTANZALUNGODIUNMESSAGGIOINITALIANODEVONOESSERENECES
TEILCIFRATODELLELETTEREEAIQUESTOEDALTRECONSIDERAZIONISIMILIPE
ODIABBASSAREILNUMERODIPERMUTAZIONIPOSSIBILICHIAVIAMMISSIBILIA
RETALECHERISULTAPRATICABILEPROVARETUTTELECHIAVICOMEABBIAMOFA
CIFRERIODICESARECOSTRUIAMOQUALCHEPROCEDURAPERVERIFICAREQUANTO
DETTO'
```

```
relfreq(m, 'A'); relfreq(m, 'E'); relfreq(m, 'I'); relfreq(m,
'O'); relfreq(m, 'N');
```

```
10.6250000000000
12.7083333333333
12.2916666666667
7.91666666666667
5.62500000000000
```

Anche le mie frasi confermano la statistica sulle frequenze delle lettere nell'Italiano. Proviamo ora a cifrare per sostituzione.

```
nm = encrypt2(m,k); nm
```

```
'PCFEOMVMEUVDUDIEIPTEUCGFNGGPCVLGMSPIVTEUCGHEZGIIGMGCUCSUHEOE
VLMUAVAEZEIVGBPECHEZGZGIIGMGFNGFUSLVEUCUFUCOMGBPGCTVLEPGZGQVI
OMVIUVAAVDIVCTVZPCRUHEPCSGDDVRREUECEIVZEVCUHGQUCUGDDGMGCGFGDI
IGEZFEOMVIUHGZZGZGIIGMGGVEBPGDIUGHVZIMGFUCDEHGMVTEUCEDESEZELC
UHEVAAVDDVMGEZCPSGMUHELGMSPIVTEUCELUDDEAEZEFNEVQEVSSEDDEAEZEV
MGIVZGFNGMEDPZIVLMVIEFVAEZGLMUQVMGIPIIGZGFNEVQEFUSGVAAEVSUOVI
FEOMGMEUHEFGDVMGFUDIMPEVSUBPVZFNGLMUFGHPMVLGMQGMEOEFVMGBPVCIU
HGIIU'
```

Se questo messaggio cifrato cade nelle mani del "nemico", che sa che il nostro cifrario è monoalfabetico, la prima cosa che farà è analizzare la frequenze delle varie lettere per individuare quelle più frequenti che corrisponderanno ad A,E,I.

```
for x in alfa:
  print(x, relfreq(nm,x))
```

```
('A', 2.70833333333333)
('B', 1.25000000000000)
('C', 5.62500000000000)
('D', 4.58333333333333)
('E', 12.2916666666667)
('F', 4.37500000000000)
('G', 12.7083333333333)
('H', 2.91666666666667)
('I', 7.08333333333333)
('L', 2.50000000000000)
('M', 6.87500000000000)
('N', 1.25000000000000)
('O', 1.66666666666667)
('P', 4.37500000000000)
```

```
('Q', 1.45833333333333)
('R', 0.625000000000000)
('S', 3.12500000000000)
('T', 1.25000000000000)
('U', 7.91666666666667)
('V', 10.6250000000000)
('Z', 4.79166666666667)
```

Dall'analisi delle frequenze ricaviamo che le lettere cifrate E,G,V, devono corrispondere alle lettere in chiaro A,E,I a meno dell'ordine. Possiamo pure ricavare che la lettera U corrisponde ad O ed una delle lettere C,I,M deve corrispondere ad N. Facendo un numero di prove non troppo elevato (18) possiamo trovare che le sostituzioni corrette sono

V->A, G->E, E->I, U->O, C->N

in quanto danno luogo ad un messaggio parzialmente decifrato che è coerente con la struttura della lingua italiana.

```
alfa = 'ABCDEFGHILMNOPQRSTUVZ '
nk =
[21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21]
nk[position('V', alfa)] = position('A', alfa)
nk[position('G', alfa)] = position('E', alfa)
nk[position('E', alfa)] = position('I', alfa)
nk[position('U', alfa)] = position('O', alfa)
nk[position('C', alfa)] = position('N', alfa)
nm; ' '; encrypt2(nm,nk)
```

```
'PCFEOMVMEUVDUDIEIPTEUCGFNGGPCVLGMSPIVTEUCGHEZGIIGMGCUCSUHEOE
VLMUAVAEZEIVGBPECHEZGZGIIGMGFNGFUSLVEUCUFUCOMGBPGCTVLEPGZGQVI
OMVIUVAAVDIVCTVZPCRUHEPCSGDDVRREUECEIVZEVCUHGQUCUGDDGMGCGFGDI
IGEZFEOMVIUHGZZGZGIIGMGGVEBPGDIUGHVZIMGFUCDEHGMVTEUCEDESEZELC
UHEVAAVDDVMGEZCPSGMUHELGMSPIVTEUCELUDDEAEZEFNEVQEVSSEDDEAEZE\
MGIVZGFNGMEDPZIVLMVIEFVAEZGLMUQVMGIPIIGZGFNEVQEFUSGVAAEVSUOVI
FEOMGMEUHEFGDVMGFUDIMPEVSUBPVZFNGLMUFGHPMVLGMQGMEOEFVMGBPVCIU
```

```
HGIIU'
' '
' N I  A IOA O  I   IONE  EE NA E    A IONE I E  E ENON O I I
A  O A I I AE  IN I E E  E E  E O  AIONO ON  E  EN A I E E A
A OA  A  AN A  N O I N E  A  IOINI A IANO E ONOE  E ENE E  A
EI  I  A O E  E E  E EEAI  E  OE A   E ON I E A IONI I I I E
IA  A  A EI N  E O I E    A IONI O  I I I  IA IA  I  I I IA
A E  E I    A  A I A I E  O A E    E E  IA I O EA  IA O A  O
E IO I E A E O    IA O  A   E  O E   A E  E I I A E  AN OA  I
O'
```

Il gruppo di lettere IONE suggerisce ZIONE che corrisponde a TEUCG e quindi ricaviamo la sostituzione T->Z

```
nk[position('T', alfa)] = position('Z', alfa)
nm; ' '; encrypt2(nm,nk)
```

```
'PCFEOMVMEUVDUDIEIPTEUCGFNGGPCVLGMSPIVTEUCGHEZGIIGMGCUCSUHEOE
VLMUAVAEZEIVGBPECHEZGZGIIGMGFNGFUSLVEUCUFUCOMGBPGCTVLEPGZGQVI
OMVIUVAAVDIVCTVZPCRUHEPCSGDDVRREUECEIVZEVCUHGQUCUGDDGMGCGFGDI
IGEZFEOMVIUHGZZGZGIIGMGGVEBPGDIUGHVZIMGFUCDEHGMVTEUCEDESEZELC
UHEVAAVDDVMGEZCPSGMUHELGMSPIVTEUCELUDDEAEZEFNEVQEVSSEDDEAEZEV
MGIVZGFNGMEDPZIVLMVIEFVAEZGLMUQVMGIPIIGZGFNEVQEFUSGVAAEVSUOVI
FEOMGMEUHEFGDVMGFUDIMPEVSUBPVZFNGLMUFGHPMVLGMQGMEOEFVMGBPVCIU
HGIIU'
' '
' N I  A IOA O  I  ZIONE  EE NA E    AZIONE I E  E ENON O I I
A  O A I I AE  IN I E E  E E  E O  AIONO ON  E  ENZA I E E A
A OA  A  ANZA  N O I N E  A  IOINI A IANO E ONOE  E ENE E  A
EI  I  A O E  E E  E EEAI  E  OE A   E ON I E AZIONI I I I E
IA  A  A EI N  E O I E    AZIONI O  I I I  IA IA  I  I I IA
A E  E I    A  A I A I E  O A E    E E  IA I O EA  IA O A  O
E IO I E A E O    IA O  A   E  O E   A E  E I I A E  AN OA  I
O'
```

Il gruppo AIONO suggerisce COMPAIONO che corrisponde a FUSLVEUCU e quindi abbiamo le sostituzioni F->C, S->M, L->P

```
nk[position('F', alfa)] = position('C', alfa)
nk[position('S', alfa)] = position('M', alfa)
nk[position('L', alfa)] = position('P', alfa)
nm; ' '; encrypt2(nm,nk);
```

```
'PCFEOMVMEUVDUDIEIPTEUCGFNGGPCVLGMSPIVTEUCGHEZGIIGMGCUCSUHEOE
VLMUAVAEZEIVGBPECHEZGZGIIGMGFNGFUSLVEUCUFUCOMGBPGCTVLEPGZGQVI
OMVIUVAAVDIVCTVZPCRUHEPCSGDDVRREUECEIVZEVCUHGQUCUGDDGMGCGFGDI
IGEZFEOMVIUHGZZGZGIIGMGGVEBPGDIUGHVZIMGFUCDEHGMVTEUCEDESEZELC
UHEVAAVDDVMGEZCPSGMUHELGMSPIVTEUCELUDDEAEZEFNEVQEVSSEDDEAEZE\
MGIVZGFNGMEDPZIVLMVIEFVAEZGLMUQVMGIPIIGZGFNEVQEFUSGVAAEVSUOVI
FEOMGMEUHEFGDVMGFUDIMPEVSUBPVZFNGLMUFGHPMVLGMQGMEOEFVMGBPVCIU
HGIIU'
' '
' NCI  A IOA O  I  ZIONEC EE NAPE M  AZIONE I E  E ENONMO I I
AP O A I I AE  IN I E E  E EC ECOMPAIONOCON  E  ENZAPI E E A
A OA  A  ANZA  N O I NME  A  IOINI A IANO E ONOE  E ENECE  A
EI CI  A O E  E E  E EEAI  E  OE A   ECON I E AZIONI IMI IPE
IA  A  A EI N ME O IPE M  AZIONIPO  I I IC IA IAMMI  I I IA
A EC E I    AP A ICA I EP O A E    E EC IA ICOMEA  IAMO A  OF
E IO ICE A ECO    IAMO  A C EP OCE   APE  E I ICA E  AN OA  I
O'
```

Dopo COMPAIONOCON riconosciamo il testo FREQUENZAPIUELEVATA che corrisponde a OMGBPGCTVLEPGZGQVIV. Ne otteniamo le sostituzioni

O->F, M->R, B->Q, P->U, Z->L, Q->V, I->T

```
nk[position('O', alfa)] = position('F', alfa)
nk[position('M', alfa)] = position('R', alfa)
nk[position('B', alfa)] = position('Q', alfa)
nk[position('P', alfa)] = position('U', alfa)
nk[position('Z', alfa)] = position('L', alfa)
nk[position('Q', alfa)] = position('V', alfa)
nm; ' '; encrypt2(nm,nk);
```

```
'PCFEOMVMEUVDUDIEIPTEUCGFNGGPCVLGMSPIVTEUCGHEZGIIGMGCUCSUHEOE
VLMUAVAEZEIVGBPECHEZGZGIIGMGFNGFUSLVEUCUFUCOMGBPGCTVLEPGZGQVI
OMVIUVAAVDIVCTVZPCRUHEPCSGDDVRREUECEIVZEVCUHGQUCUGDDGMGCGFGDI
IGEZFEOMVIUHGZZGZGIIGMGGVEBPGDIUGHVZIMGFUCDEHGMVTEUCEDESEZELC
UHEVAAVDDVMGEZCPSGMUHELGMSPIVTEUCELUDDEAEZEFNEVQEVSSEDDEAEZE\
MGIVZGFNGMEDPZIVLMVIEFVAEZGLMUQVMGIPIIGZGFNEVQEFUSGVAAEVSUOVI
FEOMGMEUHEFGDVMGFUDIMPEVSUBPVZFNGLMUFGHPMVLGMQGMEOEFVMGBPVCIU
HGIIU'
' '
'UNCIFRARIOA O  I UZIONEC EEUNAPERMU AZIONE ILE  ERENONMO IFI
APRO A ILI AEQUIN ILELE  EREC ECOMPAIONOCONFREQUENZAPIUELEVA
AINUNCIFRA OA  A  ANZALUN O IUNME  A  IOINI ALIANO EVONOE  EF
```

```
ARIAMEN EILCIFRA O ELLELE  EREEAIQUE  OE AL RECON I ERAZIONI
IMILIPERME  ONO IA  A  AREILNUMERO IPERMU AZIONIPO  I ILIC IA
I ILIA UNVALORE ALEC ERI UL APRA ICA ILEPROVARE U  ELEC IAVIC
IAMOFA  OPERILCIFRERIO ICE ARECO  RUIAMOQUALC EPROCE
URAPERVERIFICAREQUAN OA  IAMO E  O'
```

A questo punto possiamo decifrare immediatamente il messaggio ovvero determinare le sostituzioni che mancano utilizzando la nostra conoscenza della lingua italiana.

```
nk[position('A', alfa)] = position('B', alfa)
nk[position('D', alfa)] = position('S', alfa)
nk[position('H', alfa)] = position('D', alfa)
nk[position('I', alfa)] = position('T', alfa)
nk[position('N', alfa)] = position('H', alfa)
nk[position('R', alfa)] = position('G', alfa)
encrypt2(nm,nk);
```

```
'UNCIFRARIOASOSTITUZIONECHEEUNAPERMUTAZIONEDILETTERENONMODIF
APROBABILITAEQUINDILELETTERECHECOMPAIONOCONFREQUENZAPIUELEVA
FRATOABBASTANZALUNGODIUNMESSAGGIOINITALIANODEVONOESSERENECES
TEILCIFRATODELLELETTEREEAIQUESTOEDALTRECONSIDERAZIONISIMILIPE
ODIABBASSAREILNUMERODIPERMUTAZIONIPOSSIBILICHIAVIAMMISSIBILI
RETALECHERISULTAPRATICABILEPROVARETUTTELECHIAVICOMEABBIAMOFA
CIFRERIODICESARECOSTRUIAMOQUALCHEPROCEDURAPERVERIFICAREQUANT
DETTO'
```

DECIFRATO!

```

```