NEWdivcrypto_step3

April 17, 2023

Voglio presentarvi un ultimo cifrario che si basa ancora sul concetto di permutazione. Questa volta non sono le lettere che vogliamo scambiare ma sono le posizioni delle lettere. Prendiamo un messaggio abbastanza lungo.

[14]:
```
m = 'NONVOGLIOPIUSEGUIREUNALEZIONEDICRITTOGRAFIAINVITAMIA'; m, len(m)
```

[14]:
```
('NONVOGLIOPIUSEGUIREUNALEZIONEDICRITTOGRAFIAINVITAMIA', 52)
```

Abbiamo che 52 = 4*13 e decidiamo di dividere il messaggio in 4 blocchi di 13 caratteri. Questo è il principio della cosidetta "crittografia a blocchi". Anzichè cifrare lettera per lettera come negli esempi precedenti (cifrari monoalfabetici), vogliamo cifrare simultaneamente blocchi di lettere.

[24]:
```
def split(m,d):
  lm = []
  for i in range(0,floor(len(m)/d)):
    mi = ''
    for j in range(0,d):
      mi = mi + m[i*d + j]
    lm = lm + [mi]
  return lm
```

[25]:
```
d = 13; lm = split(m, d); lm
```

[25]:
```
['NONVOGLIOPIUS', 'EGUIREUNALEZI', 'ONEDICRITTOGR', 'AFIAINVITAMIA']
```

A questo punto, a ciascun blocco di caratteri possiamo applicare una permutazione dei posti occupati dalle lettere. Questa permutazione costituirà la chiave del crittosistema.

[26]:
```
def blockperm(b,k):
  d = len(b)
  nb = ''
  for i in range(0,d):
    nb = nb + b[k[i]]
  return nb
```

Facciamo l'esempio di un singolo blocco.

[18]:
```
b = 'bibliotecario'; k = [0,7,9,6,12,8,5,1,3,4,2,10,11];
len(b), len(k)
```

[18]:
```
(13, 13)
```

[19]:
```
blockperm(b,k)
```

[19]:
```
'beatocoilibri'
```

Bello! Questo è un famoso anagramma, ovvero una parola di senso compiuto che si ottiene da una altra parola per permutazione delle lettere. Naturalmente, una permutazione a caso non avrebbe prodotto una parola del vocabolario.

[20]:
```
k = [3, 12, 9, 0, 1, 6, 5, 8, 4, 2, 10, 7, 11]
blockperm(b,k)
```

[20]:
```
'loabitocibrei'
```

La nostra funzione di cifratura si otterrà applicando la stessa permutazione di posti ad ogni blocco di caratteri.

[27]:
```
def encrypt3(m,k):
  d = len(k)
  lm = split(m,d)
  nm = ''
  for mi in lm:
    nm = nm + blockperm(mi,k)
  return nm
```

[29]:
```
print(m)
print(k)
print(encrypt3(m,k))
```

```
NONVOGLIOPIUSEGUIREUNALEZIONEDICRITTOGRAFIAINVITAMIA
[3, 12, 9, 0, 1, 6, 5, 8, 4, 2, 10, 7, 11]
VSPNOLGOONIIUIILEGUEARUENZDRTONRCTIEOIGAAAAFVNTIIMII
```

[31]:
```
import random
d = 13
k = random.sample(range(0,d),d)
mm = encrypt3(m,k)
m, k, mm
```

[31]:
```
('NONVOGLIOPIUSEGUIREUNALEZIONEDICRITTOGRAFIAINVITAMIA',
 [7, 0, 5, 11, 1, 8, 12, 10, 2, 4, 3, 9, 6],
 'INGUOOSINOVPLNEEZGAIEURILUIOCGNTROEIDTRIANIFTAMIIAAV')
```

Richiamiamo un paio di funzioni che ci sono servite in precedenza. Servano qui per ottenere la funzione di decifratura che si ottiene semplicemente come cifratura mediante la permutazione inversa.

[33]:
```
def position(x,l):
  i = 0
  y = l[i]
  while y != x:
    i = i + 1
    y = l[i]
  return i

def invperm(p):
  np = [position(i,p) for i in range(0,13)]
  return np
```

[34]:
```
def decrypt3(m,k):
  nk = invperm(k)
  nm = encrypt3(m,nk)
  return nm
```

[36]:
```
k = random.sample(range(0,d),d)
mm = encrypt3(m,k)
k, m, mm, decrypt3(mm,k)
```

[36]:
```
([5, 1, 3, 9, 12, 8, 0, 4, 7, 11, 10, 6, 2],
 'NONVOGLIOPIUSEGUIREUNALEZIONEDICRITTOGRAFIAINVITAMIA',
 'GOVPSONOIUILNEGILIAERNZEUUCNDTRTOIIGORENFAAATAIIIMVI',
 'NONVOGLIOPIUSEGUIREUNALEZIONEDICRITTOGRAFIAINVITAMIA')
```

I moderni sistemi di crittografia (a blocchi) come AES - Advanced Encryption Standard (2002 - ??) si basano essenzialmente su una ripetizione di cifrature ottenute mediante permutazioni di lettere e posti. Altri sistemi (chiave pubblica-privata) si basano invece su proprietà dei numeri interi. Come vedete si tratta di Matematica!

Nuova matematica, tipi di algoritmi e computer sempre più potenti si stanno affacciando però sull'eterna lotta fra chi cerca di cifrare in modo sempre più sicuro e chi tenta di decifrare in maniera sempre più intelligente. Con le vostre giovani menti, anche voi potreste partecipare se lo vorrete, a questa evoluzione tecnologica!

Volete fare qualche esercizio-gioco? Proviamo a decifrare qualche messaggio.

Cifrario a scorrimento:

tpopqpsdlrvftulspnbol

odpdzhpdznfdvhubh

Cifrario a permutazione di posti:

ieistcmoi (un singolo blocco)

roeoirisppisoarimfst (due blocchi di lunghezza 10)

[37]:
```
alfa = 'abcdefghilmnopqrstuvz'

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
```

[38]:
```
m = 'aaaaaaaaaaa'
encrypt(m, 'z')
```

[38]: 'zzzzzzzzzzz'

[42]:
```
m = 'xxxxxxxxxxxxxxx'
print(len(m))
d = floor(len(m)/2)
k = random.sample(range(0,d),d)
print(k)
encrypt3(m, k)
```

```
16
[1, 3, 5, 2, 4, 7, 6, 0]
```

[42]: 'xxxxxxxxxxxxxxx'

[34]:
```

```

[34]: