88-CRIPT-one-time-pad

March 5, 2018

```
In [1]: alf = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ:;,#$@'

In [2]: len(alf)

Out[2]: 32
```

£Por qué queremos usar un alfabeto de 32 letras?

```
In [3]: texto = 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCO\
        UNTINGCALCULATIONMEASUREMENTANDTHESYSTEMATICSTUDYOFTHESHAPESANDMOTIONSOF\
        PHYSICALOBJECTSPRACTICALMATHEMATICSHASBEENAHUMANACTIVITYFORASFARBACKASWR\
        ITTENRECORDSEXISTRIGOROUSARGUMENTSFIRSTAPPEAREDINGREEKMATHEMATICSMOSTNOT\
        ABLYINEUCLIDSELEMENTSMATHEMATICSDEVELOPEDATARELATIVELYSLOWPACEUNTILTHERE\
        NAISSANCEWHENMATHEMATICALINNOVATIONSINTERACTINGWITHNEWSCIENTIFICDISCOVER\
        IESLEDTOARAPIDINCREASEINTHERATEOFMATHEMATICALDISCOVERYTHATCONTINUESTOTHE\
        PRESENTDAY'

In [4]: len(texto)

Out[4]: 513
```

Apartado 1 : generar clave

```
In [5]: clave = [randint(0,1) for muda in xrange(10^6)]

In [6]: [sum(clave[k*10^5:(k+1)*10^5]) for k in srange(10)]

Out[6]: [50037, 50079, 50190, 49906, 49901, 49788, 50125, 49932, 50148, 50045]

In [7]: def cadena(L):
            C = ''
            for item in L:
                C += str(item)
            return C

In [8]: clave_c = cadena(clave);clave_c[0:10]

Out[8]: '0000010000'
```

Apartado 2: Codificar

```
In [9]: L_alf = list(alf)

In [10]: def ord2(c):
             return L_alf.index(c)

In [11]: def chr2(n):
             return L_alf[n]

In [12]: def codificar(texto):
             texto_c = ''
             L = map(ord2,list(texto))
             for item in L:
                 L1 = ZZ(item).digits(base=2,padto=5)
                 L1.reverse()        ##los digitos estan en el orden contrario al que queremos
                 texto_c += cadena(L1)
             return texto_c

In [13]: def descodificar(texto_e):
             texto_d = ''
             if len(texto_e) == 0:
                 return texto_d
             texto_dp = descodificar(texto_e[5:])
             return chr2(ZZ(texto_e[:5],base=2))+texto_dp

In [ ]: COD = codificar(texto)

In [14]: descodificar(COD)

Out[14]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULA
         EASUREMENTANDTHESYSTEMATICSTUDYOFTHESHAPESANDMOTIONSOFPHYSICALOBJECTSPRACTICALMATHEMA
         ASBEENAHUMANACTIVITYFORASFARBACKASWRITTENRECORDSEXISTRIGOROUSARGUMENTSFIRSTAPPEAREDIN
         MATHEMATICSMOSTNOTABLYINEUCLIDSELEMENTSMATHEMATICSDEVELOPEDATARELATIVELYSLOWPACEUNTIL
         NAISSANCEWHENMATHEMATICALINNOVATIONSINTERACTINGWITHNEWSCIENTIFICDISCOVERIESLEDTOARAPI
         EASEINTHERATEOFMATHEMATICALDISCOVERYTHATCONTINUESTOTHEPRESENTDAY'
```

Apartado 3: encriptar

```
In [15]: def suma_s(c1,c2):
             if (c1 == '0' and c2 == '0') or (c1 == '1' and c2 == '1'):
                 return '0'
             else:
                 return '1'

In [16]: def encriptar(texto_c,clave):
             texto_e = ''
             for int in srange(len(texto_c)):
                 texto_e += suma_s(texto_c[int],clave[int])
             return texto_e
```

```
In [17]: texto_c = codificar(texto);texto_c

Out[17]: '1001100111100010111010100001100011110011001110010010100100100010001110001010000000001
         0100111000100000000101001101000011100110100000011010001101011011100011001000000100000
         1100010010000000100100111001101010000110100110011000000010011001110010001100000001001
         0000101001000011001001010100100010111011100111100100000110010110001011100110000010011
         0011011001101000011010011000010000000101100010101000101100000100110100001110011010110
         0000001001010100100010010001100001000110110011000000110100011100110011100100100101100
         0100110010001100000001001101000000101001010011101000001111000011100010110011001110010
         0001110000001111001001001000000011010001101100011101001101000011100110110010011100010
         1001111100010010010000001000000010110111000001010010010000010100111001001111100010000
         0100110100000010000000101101100000001001100111001000110000000100110100000010100100011
         0100100000100100001000110100000001111010001100000000110100000000101001101000101010100
         1110000010101110100010000010010001010000010001000010000000010010100000010010101101000
         0100111001100100011011000100100000100111010001000111001000100101110100010010100111000
         0001100111010001011101010010010000001000100110101000110000100011011001110010001010100
         1100101001100000011110111100100000001000100100000011010000110100110100010010000100010
         0000001001100111001000110000000100110100000010100100110001110100101001101101011101001
         0000010101111000010000110100100101000001001011010000001110010001000101100100011000010
         1100111001001100000001001100111001000110000000100110100000010100100000110010010101001
         1011100111100100000110000010011000001000100100010110000010011010001010100100010111100
         0010110111010110011110000000010001001010001101100110100001011100110011100100100010010
         1000000100010010100100000001101000100010010110001110010001101011000000010011001110010
         0000001001101000000100000001011010000110101101011101010100000100110100001110011011001
         0011011001100100100010000000010100110100001101001101011001000100110011101101001001011
         0000100100000100011011001101000001010100000010000110100010010000100111010101001001000
         0001001001001011001000001110011011100000010001000000111101000000110100001101000101000
         0000001001000100010000110110011001110010010001000001001100100011100010101100000001001
         1001000110000000100110100000010000000101100011010001001000010011101010100100100011100
         1001110000010011000100111001101100110100001101101000010010010100110111010011001110010
         1100010010010010001000110110011000110000011000'

In [18]: texto_en = encriptar(texto_c,clave_c);texto_en

Out[18]: '0111111011010101000001010101101010101100010001000000101111011001101110000011011111000
         0011110111000100001101111101101000110111110000000000000000011010001000011110101110111
         1001000111011011011101100010101100100111010100111111110000100110111011110101100000011
         1011011010000111100001101011110110001001010100110000101001011011100000011011011101001
         0110001001110001110110000000011101011001001000010110110110110010011111111011001111000
         1001111011000110101101100011110111010011001100001100101111111101100000000010011101001
         1100111111000100000111000101000000101011001001001011001011111001001001000010111110101
         1110001000011110101010001101001000100111101101110000001011100110101101001010111110101
         0010011010000010011101101111011101100111111101100001010111100100101101010000110111111
         0100011011001100011010110111010010000001001101100000110001100100001000000110110111111
         0011100101110111010000101101011110000001110101100010011000011101010111000111101101101
         0001000111110100000011101011100100010111000000100101110110111111110000110001111101000
         1000111001011011001110010101010111101110001010100110101111000000000000001101101011101
         1010001101010101011010011101011100110001111101101001011011010000110001011101011001110
```

```
01100110010111101001010101110100110000110001100001001101001101011110110100101101100000
00001011011101100001000011110111101110010001011101001000000011100101110011110011111010
10100111010011011100110111011001011000110000001110111101111011110010010010000101011001
01111101111011100001011001001000001101010011011100110010001101010010110000100010101010
11101110011010010111000001010111100101101010100001011111110110111001110001001111000001
11010111100000010000100010001011010010011010100111001100000001010001001111110010100100
10110001110101101111000111100000110000010110101011100110110011001001111111110111111001
11111010010000110111011111100110010110011111000101110011001101100001111100000011001011
11100110111111001101010010101001000101000111001001110011100111001100101101001101001111000110
00001011011011011010000010000011111111101001000111010011011100011011101010011010111011
10110110010001011001000000100011110000011110001101010110110000010100000000001001000101
01000101110110110101110001011001100111010011101011101000000100001001010101011001001010
01001110100001010001100101110101001100111101101111101100100001011001001010100110010011
10011011011001101001010101111101110110011011000100111001000110101110010101111000100110
10011010000011000000110110001000101010000001001'
```

```
In [19]: descodificar(texto_en)
```

```
Out[19]: 'P;KQKWVMIQF#TOBXYUPOEG@NDPQAAGRB#O#LEO;OYVSNJ@YJXPLAN;NUHQ:$YSUYKLOA;OSOYTR;ADVSILNW
         FHWGWY$#GMGL@WACOSZT$EDRICWJFS@ESC@FBYQ$VDJCPNYC,::K@KSJUCO;;WP#QV,S:Q;@MRWMNN:ICNQMM
         ,OLXILLYDVRGDVOHW;SEPUB:,ROAS#X$DD#BZDS;HFK$,KTLYAANV;TI:VNHLTD#UW:DC#M#SZS$SV:MGGCNG
         QC;WCD;;SF:IBZOPH:ZJ:NZXMWGA###,SIKZG@POCZEDKNZSGUWCFKV;TJOBLZNKC@;PHCPB;V,BBCFUTKOMA
         NMOW$HQMC:XGZSP@PZP$SDO@TFT,LTGYPQGLTZX,:SURI,TTTFU:PDWC;NUCB@#EOTOG#JV;PNSFSAR,DY:WY
         GRO;LRMZ:OXICCKVSKYTUFDF:THW@MQWJKMTTG;GSV$#TMJZDLSXRG#GQMBWEKQJ'
```

```
In [20]: encriptar(encriptar(texto_c,clave_c),clave_c)==texto_c
```

```
Out[20]: True
```

```
In [21]: descodificar(encriptar(encriptar(texto_c,clave_c),clave_c))
```

```
Out[21]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULA
         EASUREMENTANDTHESYSTEMATICSTUDYOFTHESHAPESANDMOTIONSOFPHYSICALOBJECTSPRACTICALMATHEMA
         ASBEENAHUMANACTIVITYFORASFARBACKASWRITTENRECORDSEXISTRIGOROUSARGUMENTSFIRSTAPPEAREDIN
         MATHEMATICSMOSTNOTABLYINEUCLIDSELEMENTSMATHEMATICSDEVELOPEDATARELATIVELYSLOWPACEUNTIL
         NAISSANCEWHENMATHEMATICALINNOVATIONSINTERACTINGWITHNEWSCIENTIFICDISCOVERIESLEDTOARAPI
         EASEINTHERATEOFMATHEMATICALDISCOVERYTHATCONTINUESTOTHEPRESENTDAY'
```

Apartado 4: Discutir la seguridad del sistema

Este sistema es totalmente seguro siempre que se consiga mantener en secreto la clave, ésta se haya generado realmente de manera aleatoria y no se reutilice nunca un trozo de la clave ya usado. Pensemos en un ataque por "fuerza bruta": Si el mensaje tiene N bits tendríamos que probar las 2^N posibles claves y, para cada una de ellas, obtendríamos un posible mensaje, pero TODOS los mensajes de N bits aparecerían y no habría manera de decidir entre ellos. Por ejemplo, para una cierta clave podríamos obtener "ATACAMOS", pero para otra obtendríamos "RETIRADA" y el mensaje encriptado no contiene ninguna información que nos permita decidir entre los dos.

Podemos argumentar también lo siguiente:

En el sistema de César todas las letras se encriptan con la misma clave y el mensaje encriptado contiene todavía un montón de información, las frecuencias, acerca del mensaje original. El sistema también permite un ataque de fuerza bruta y es muy vulnerable.

En el sistema de Vigenere las letras cuya posición difiere en k unidades, la longitud de la clave, se encriptan igual. El mensaje encriptado, si es de suficiente longitud, todavía contiene suficiente información como para que sea posible un análisis de frecuencias. Un ataque por fuerza bruta es posible en teoría, pero si la clave es muy larga imposible en la práctica.

El sistema que estamos discutiendo es, esencialmente, como un Vigenere con la longitud de la clave igual a la longitud del texto. Si la clave se genera aleatoriamente, no queda en el mensaje encriptado ninguna información de frecuencias relevante. Si la clave se reutiliza el sistema ya es como un Vigenere, con un texto de longitud el doble de la clave, y el mensaje encriptado contiene algo de información sobre el original que puede ser utilizada.

Apartado 5 : Variante

£Podemos usar una suma llevando, es decir $1+1=10$ en lugar de $1+1=0$? Para que este método funcione es conveniente que el mensaje encriptado contenga también el número de bits del mensaje original, ya que es posible que el mensaje encriptado sea más largo porque sumamos llevando. Sin embargo, la longitud del mensage encriptado sólo puede ser igual a la del mensaje original o un bit mayor, de forma que si no conocemos la longitud del original podemos hacer dos pruebas y ver cuál funciona.

Como queremos sumar llevando parece mejor efectuar la suma en base 10 en lugar de implementar una suma binaria llevando.

```
In [22]: def encriptar2(texto_c,clave_c):
             N1 = ZZ(texto_c,base=2)
             N2 = ZZ(clave_c[:len(texto_c)],base=2)
             return len(texto_c),cadena((N1+N2).digits(base=2))

In [23]: def desencriptar2(texto_e,clave_c):
             L = list(texto_e[1])
             L.reverse()
             N1 = ZZ(join(L,sep=''),base=2)
             N2 = ZZ(clave_c[:texto_e[0]],base=2)
             C = cadena((N1-N2).digits(base=2,padto=texto_e[0]))
             L = list(C)
             L.reverse() #Otra vez, digits produce el orden inverso al que queremos
             return cadena(L)

In [24]: desencriptar2(encriptar2(codificar(texto),clave_c),clave_c)==codificar(texto)

Out[24]: True

In [25]: C = join([alf[randint(0,31)] for j in srange(10^5)],sep='')
         desencriptar2(encriptar2(codificar(C),clave_c),clave_c)==codificar(C)

Out[25]: True
```