In [2]:
```
F=GF(11); F
```

Out[2]:
```
Finite Field of size 11
```

In [3]:
```
n=10; k=5;
```

In [5]:
```
C=codes.ReedSolomonCode(F,n,k); C #No hacer en casa: usar un RS no es sse
guor en la práctica, lo hacemos aquí por motivos académicos
```

Out[5]:
```
[10, 5, 6] Reed-Solomon Code over GF(11)
```

In [6]:
```
t=floor(((C.minimum_distance()-1)/2)); t
```

Out[6]:
```
2
```

In [7]:
```
G=C.generator_matrix(); G #una matriz generadora de C, es privado
```

Out[7]:
```
[ 1  1  1  1  1  1  1  1  1  1]
[ 1  2  4  8  5 10  9  7  3  6]
[ 1  4  5  9  3  1  4  5  9  3]
[ 1  8  9  6  4 10  3  2  5  7]
[ 1  5  3  4  9  1  5  3  4  9]
```

In [8]:
```
permutacion=[6,5,3,4,2,8,1,9,7,0]; permutacion
```

Out[8]:
```
[6, 5, 3, 4, 2, 8, 1, 9, 7, 0]
```

In [11]:
```
P=matrix(F,n,n);
for i in range(10):
    P[i,permutacion[i]]=1;
P # matriz permutacion
```

Out[11]:
```
[0 0 0 0 0 0 1 0 0 0]
[0 0 0 0 0 1 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0]
[0 0 1 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 1 0]
[0 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 1 0 0]
[1 0 0 0 0 0 0 0 0 0]
```

In [12]:
```
S=random_matrix(F,k,k); S #matriz scramble
```

Out[12]:
```
[ 0  1  5  0  5]
[ 9  1  2  0  8]
[ 8  3  6 10  2]
[ 2  7  7 10  4]
[ 4  2  7  4  8]
```

In [14]:
```
S.determinant() #¿Es invertible?
```

Out[14]:
```
5
```

In [15]:
```
S^(-1) #como el determinante es distinto de cero, es invertible
```

Out[15]:
```
[ 5 10  5  3  2]
[10  1  6  1 10]
[ 2  3  8  0  2]
[ 4  0  3  9  6]
[ 5 10  4  2  7]
```

In [16]:
```
Gprima = S * G * P; Gprima
```

Out[16]:
```
[ 0 10 10  0  7  3  0  2  9  3]
[ 5  0  4  3  1  4  9  7  7  6]
[ 0  0  0  3  0  7  7  8  3  8]
[ 6  0  2  2 10  1  8  9  7  8]
[ 5  3  2  8  7  9  3  4  2  8]
```

In [17]:
```
(Gprima,t) #clave pública: Gprima y la capacidad correctora
```

Out[17]:
```
(
[ 0 10 10  0  7  3  0  2  9  3]
[ 5  0  4  3  1  4  9  7  7  6]
[ 0  0  0  3  0  7  7  8  3  8]
[ 6  0  2  2 10  1  8  9  7  8]
[ 5  3  2  8  7  9  3  4  2  8], 2
)
```

In [18]:
```
(G,S,P,'Algoritmo de decoodificación') #clave privada
```

Out[18]:
```
(
[ 1  1  1  1  1  1  1  1  1  1]  [ 0  1  5  0  5]
[ 1  2  4  8  5 10  9  7  3  6]  [ 9  1  2  0  8]
[ 1  4  5  9  3  1  4  5  9  3]  [ 8  3  6 10  2]
[ 1  8  9  6  4 10  3  2  5  7]  [ 2  7  7 10  4]
[ 1  5  3  4  9  1  5  3  4  9], [ 4  2  7  4  8],

[0 0 0 0 0 0 1 0 0 0]
[0 0 0 0 0 1 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0]
[0 0 1 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 1 0]
[0 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 1 0 0]
[1 0 0 0 0 0 0 0 0 0], 'Algoritmo de decoodificación'
)
```

In [20]:
```
#ya tenemos las claves, Bob puede mandar un mensaje a Alice usando la cla
ve pública
```

In [22]:
```
m=vector(F,[5,9,5,5,6]); m #mensaje de longitud k a enviar codificado, me
nsaje en claro
```

Out[22]:
```
(5, 9, 5, 5, 6)
```

In [23]:
```
x=m*Gprima;x
```

Out[23]:
```
(6, 2, 9, 1, 4, 2, 9, 6, 5, 10)
```

In [25]:
```
c=x; c[0]=c[0]+5; c[9]=c[9]+7; c #mensaje cifrado que Bob envia a Alice
```

Out[25]:
```
(0, 2, 9, 1, 4, 2, 9, 6, 5, 6)
```

In [26]:
```
#Alice recibe c y procede a descifrar
```

In [27]:
```
cprima = c * P^(-1); cprima
```

Out[27]:
```
(9, 2, 1, 4, 9, 5, 2, 6, 6, 0)
```

In [29]:
```
mprimaporG = C.decode_to_code(cprima); mprimaporG
```

Out[29]:
```
(9, 2, 1, 4, 9, 5, 2, 10, 6, 6)
```

In [30]:
```
mprima= C.decode_to_message(cprima); mprima #mejor sacamos el vector info
rmación directamente
```

Out[30]:
```
(1, 10, 7, 3, 10)
```

In [32]:
```
mprima*G == mprimaporG
```

Out[32]:
```
True
```

In [33]:
```
mprima*S^(-1) #recuperamos el mensaje enviado por Bob
```

Out[33]:
```
(5, 9, 5, 5, 6)
```

In [34]:
```
#ataque genérico de descodificación con un conjunto de información
```

In [36]:
```
ck=c[1:6]; ck #Eve seleccciona las posiciones 2 a 6 suponiendo que no tie
nen errores
```

Out[36]:
```
(2, 9, 1, 4, 2)
```

In [37]:
```
Gprimak = Gprima[0:,1:6]; Gprimak
```

Out[37]:
```
[10 10  0  7  3]
[ 0  4  3  1  4]
[ 0  0  3  0  7]
[ 0  2  2 10  1]
[ 3  2  8  7  9]
```

In [39]:
```
ck*(Gprimak)^(-1) #recuperamos m porque la matriz es invertible en este c
aso
```

Out[39]:
```
(5, 9, 5, 5, 6)
```

In [ ]:
```

```