

Códigos y criptografía: Curso 2022-2023

Práctica 2: Cifrado Hill.

Cifrado de permutación (caso particular del Hill)

- Se usarán las siguientes funciones de la práctica 1:

letter_number e *inv_module*.

1. Función *hill_cipher*

```
1 function code=hill_cipher(A,text)
```

Se trata de una función que cifra un texto a partir de una matriz A mediante el cifrado Hill.

Entradas:

A: la matriz que va a ser la clave.

text: el texto claro.

Salida: el criptograma.

Ejemplos:

```
1 >> code=hill_cipher ( [2 1 3;2 1 1;0 4 1] , 'en un lugar de cordoba')
2 code = dosselmdrpxmdxueee
```

```
1 >> code=hill_cipher ( [2 1 3;2 1 1;0 4 1] , 'un lugar de cordoba')
2 code = hmjuuuxppskxxvhlsh
```

- NOTA: Ahora correspondería preparar una función para descifrar un criptograma obtenido mediante un cifrado Hill, pero, ¿hace falta hacerla?

2. Función *permutation*

```
1      function permute=permutation(p)
```

Se trata de una función que comprueba si el vector de entrada representa una permutación de $\{1, 2, \dots, n\}$, siendo n el número de elementos del vector.

Entrada: el vector que queremos comprobar si es o no una permutación.

Salida:

permute = 0 si el vector no es una permutación.

permute = 1 si el vector es una permutación.

Ejemplos:

```
1      >> permute=permutation([5 4 3 1 2])
2      permute = 1
```

```
1      >> permute=permutation([5 4 6 1 2])
2      permute = 0
```

3. Función *matrix_per*

```
1      function matrix=matrix_per(p)
```

Se trata de una función que comprueba si la entrada es una permutación, y en ese caso construye la matriz asociada a ella.

Entrada: un vector, que supuestamente debe ser una permutación debiendo comprobando la función que lo es.

Salida: la matriz asociada a la permutación en caso de que la entrada lo sea, o un mensaje de error en caso contrario.

Ejemplos:

```
1      >> matrix=matrix_per ([2 1 4 5 3])
2      matrix =
3          0  1  0  0  0
4          1  0  0  0  0
5          0  0  0  1  0
6          0  0  0  0  1
7          0  0  1  0  0
```

```
1      >> matrix=matrix_per ([2 1 4 5 6])
2      The input is not a permutation.
```

4. Función *cipher_permutation*

```
1 function code=cipher_permutation(p,text)
```

Se trata de una función que cifra un texto a partir de una permutación aplicando el cifrado Hill en caso de ser posible. En otro caso muestra un mensaje de error.

Entradas:

p: el vector que debe ser una permutación. La función debe comprobarlo.

text: el texto llano a cifrar.

Salida: el criptograma usando Hill y la permutación, si es posible realizar el cifrado.

Ejemplo:

```
1 >> code = cipher_permutation ([2 4 6 5 3 1], 'hola me voy de puente')
2 cifrado = oaemlhodpeyvetwenu
```

5. Función *decipher_permutation*

```
1 function text=decipher_permutation(p,code)
```

Se trata de una función que descifra un criptograma conociendo la permutación utilizada como clave.

Entradas:

p: el vector que debe ser una permutación. La función debe comprobarlo.

code: el criptograma.

Salida: el texto llano.

Ejemplo:

```
1 >> text = decipher_permutation ([6 4 2 1 3 5], 'eaohlmvdovyeoccaaiyhens')
2 text = holamevoydevacacioneshoy
```

6. Función *crypto_hill*

```
1 function matrix=crypto_hill(text,code,d)
```

Se trata de una función que halla, si es posible, la matriz clave empleada para el cifrado hill conociendo parte del texto llano, del criptograma y el orden de la matriz. El criptoanálisis se realiza con un ataque de tipo Gauss-Jordan.

Entradas:

text: el fragmento inicial del texto llano, de longitud al menos d^2 .

code: el fragmento inicial del criptograma, de longitud al menos d^2 .

d: el tamaño de los bloques, es decir, el orden de la matriz que estamos suponiendo para el cifrado Hill.

Salida: la matriz clave del cifrado, si es posible obtenerla.

Ejemplo:

```
1  >> text='lasaparienciasengañan';
2  >> code='vovqldhlgzxivñoihccsañdh';
3  >> matrix=crypto_hill(text,code,4)
4
5  matrix =
6      1     3     2     5
7      7    24     1     1
8      0     0     4     2
9      3     5     2     1
```

- **IMPORTANTE:** Hay que prestar mucha atención al preparar las matrices correspondientes al texto claro y al texto cifrado en la función *crypto_hill*:
 - ¿sirve cualquier longitud de texto?
 - ¿qué ocurre si las dos cadenas no tienen la misma longitud?
 - ¿y si la longitud no es un múltiplo de d ?