Códigos y Criptografía - Curso 2019-2020

Práctica 4: Cifrado con Mochila y

Cifrado con Mochila Trampa

- Mientras no se diga lo contrario, si es necesario asignar números a letras se hará uso del código ASCII.

1.- Función valida=mochila (s)

Función para analizar si un vector fila es una mochila simple o supercreciente. Se trata de una función de control, previa al cifrado de tipo mochila.

Entrada: vector fila, que representa la mochila. La función debe comprobar si realmente representa una mochila, y en caso afirmativo si ésta es simple o supercreciente.

Salida: *valida=0* si la mochila no es supercreciente

valida=1 si la mochila es supercreciente

Ejemplo

>> valida=mochila([5 6 7 2])

la mochila no es supercreciente, sus elementos no estan ordenados en orden creciente

valida = 0

>> valida=mochila([5 6 7 20])

la mochila no es supercreciente

valida = 0

>> valida=mochila([5 6.5 7 20])

algun valor de la mochila no es entero positivo

valida = 0

>> valida=mochila([5 6 17 40])

valida = 1

2.- Función v=sol_mochila (s,obj)

Función para comprobar si una mochila dada cumple un determinado objetivo con el algoritmo estudiado para mochilas supercrecientes. Es una segunda función de control para poder implementar el cifrado de tipo mochila.

Entradas:

s: mochila. La función debe comprobar si es realmente una mochila.

obj: objetivo a alcanzar. La función debe comprobar si la entrada cumple los requisitos para poder ser un objetivo

Salida: En caso de que el objetivo se cumpla, un vector *v* indicando los valores de la mochila que permiten obtenerlo. En caso contrario *v=0* y un mensaje. Obsérvese que el objetivo pudiera alcanzarse aún cuando la mochila no fuera supercreciente. Por tanto, también debe indicar si la mochila es supercreciente o no.

Ejemplo

>> v=sol_mochila([20 7 4 1],12)

la mochila no es supercreciente, sus elementos no estan ordenados en orden creciente

v = 0 1 1 1

>> v=sol_mochila([20 5 736 13 2],35)

la mochila no es supercreciente, sus elementos no estan ordenados en orden creciente

con el algoritmo usado no encuentro el objetivo (la mochila no es supercreciente)

v = 0

>> v=sol_mochila([4 10 20 47 100],53)

el objetivo no se alcanza, la mochila si es supercreciente

v = 0

>> v=sol_mochila([4 10 20 47 100],71)

la mochila es supercreciente

v = 1 0 1 1 0

Para seguir adelante pudieran ser convenientes las funciones char, abs('texto'), dec2bin, bin2dec, bitget e int2str.

3.- Función cifrado=cifr_mochila (s, texto)

Función para cifrar un mensaje con una mochila (no tiene porqué ser supercreciente).

Entradas:

s: mochila. La función debe comprobar si es realmente una mochila.

texto: texto a cifrar.

Salida: vector numérico que se corresponda con el mensaje cifrado.

Ejemplo

```
>> cifrado=cifr_mochila([2 4 10 19 40],'hola')
cifrado = 54    40    71    31     6     6    73

>> cifrado=cifr_mochila([2 4 10 19 40.2],'hola')
algun valor de la mochila no es entero positivo
cifrado = 0
```

4.- Función cifrado=cifroMochilaSuper (s,texto)

Función que se asegura de que la mochila introducida es supercreciente, y sólo en ese caso cifra el texto.

Entradas:

s: vector que debe ser una mochila supercreciente. La función debe comprobarlo.

texto: el texto llano a cifrar.

Salida: vector numérico que se corresponda con el mensaje cifrado en caso de ser la mochila supercreciente.

Ejemplo

```
>> cifrado=cifroMochilaSuper([2 5 10 23 56], 'hora de comer')
cifrado = 71  56  91  91  10   7  61   0  15  58  25  25   0   7  86  94  71  68  25  91  66

>> cifrado=cifroMochilaSuper([2 5 10 23 36], 'hora de comer')
la mochila no es supercreciente
cifrado = [ ]
```

5.- Función texto=des_mochila (s,cifrado)

Función para descifrar un texto cifrado conociendo la mochila supercreciente que se ha utilizado como clave.

Entradas:

s: mochila que debe ser supercreciente.

cifrado: vector de cifrado.

Salida: El texto claro.

Ejemplo

>> s=[2 5 10 23 56];

>> cifrado=[71 56 91 91 10 7 61 0 15 58 25 25 0 7 86 94 71 68 25 91 66];

>> texto=des_mochila(s,cifrado)

la mochila es supercreciente

texto = 'hora de comer'

>> s=[2 5 10 23 36]

>> texto=des_mochila(s,cifrado)

la mochila no es supercreciente

no podemos asegurar la unicidad del descifrado, por lo tanto no lo hacemos

texto = [ ]

Pasamos a continuación al cifrado asimétrico con Mochila Trampa.

6.- Función factores_c=factorescomunes (w , s)

Función para comprobar si un número w tiene factores primos comunes con los elementos de la mochila s, porque en caso de tener no será un buen número que actúe como factor entre la mochila simple y la mochila trampa.

Entradas:

w: un número entero.

s: una mochila. Aunque para el cifrado deberá ser supercreciente, para implementar esta función no debe serlo necesariamente.

Salida: factores_c=0 si no hay factores comunes.

factores_c=1 si hay factores comunes, junto con un mensaje que indique con que elementos de la mochila tiene factores comunes.

Ejemplo

```
>> factores_c=factorescomunes(47,[5 6 81 345 634])
factores_c = 0

>> factores_c=factorescomunes(48,[5 6 81 345 634])
los numeros de la mochila con factores comunes a w son
ans =  6    81   345   634
factores_c = 1
```

7.- Función [cpubl, cpriv]=mochila_mh (s)

Función que permita al usuario (receptor) generar una clave privada y una clave pública adecuada a partir de una mochila supercreciente

Entrada: mochila supercreciente. La función debe asegurarse que el vector es una mochila supercreciente.

Salidas:

cpubl: mochila trampa creada a partir de *s* y de los valores *mu* y *w*. Para ello:

- El valor *mu* debe introducirlo el usuario, por lo que se lo debe pedir la función. Debe ser adecuado, para ello basta con que sea superior al doble del último elemento de *s*.

- El valor *w* lo buscara la función asegurándose que tenga inverso módulo *mu* y que no tenga factores comunes con los elementos de *s*.

cpriv: un vector fila de dos elementos: *mu* y el inverso de *w* módulo *mu*.

Ejemplo

```
>> [cpubl,cpriv]=mochila_mh([2 5 8 23 67 131])
necesitamos  un entero MAYOR QUE   236
escribe un entero  que cumpla esa condicion
531

cpubl =  523   511   499   439   263     7
cpriv = 531   398
```

¿Hace falta una función nueva para cifrar un texto a partir de una mochila trampa? ¿Por qué?

8.- Función texto=des_mmh (s, cifrado, mu, invw)

Función que permita al usuario (receptor) descifrar un mensaje cifrado con su clave pública asociada.

Entradas:

s: la mochila supercreciente.

cifrado: el texto numérico recibido que representa el texto cifrado.

mu, invw: los elementos de la clave privada (obtenidos a partir de la función anterior).

Salida: el texto claro.

Ejemplo

```
>> s=[2 5 8 23 67 131]

>> cifrado=[1712  1213  439  523  1449  1736  1979  1724  499  702
1041  530  1449  1297  0  1297  499  702  518  969  499  702
1041  530  499  709  518  530  1449  1225  518  969]

>> texto=des_mmh(s,cifrado,531,398)

texto = 'ya son las 2 de la tarde'
```

Por último, vamos a implementar el criptoanálisis de Shamir y Zippel. Como se ha estudiado en teoría, se supone conocida la mochila difícil o trampa, el módulo de trabajo, y supondremos que los dos primeros elementos de la mochila tienen inverso módulo *mu*.

9.- Función s=cripto_shamir_zippel (cpubl, mu)

Función que permita criptoanalizar una mochila trampa conocido el módulo de trabajo (no tendremos en cuenta que los los elementos de la mochila estén ordenados y los dos primeros tengan inverso módulo *mu*).

Entradas:

cpubl: la mochila trampa.

mu: el módulo de trabajo.

Salida: la mochila supercreciente asociada.

El programa (o función) debe indicar el rango en el que está buscando el posible primer elemento de la mochila supercreciente, y cada vez que se acabe el rango pedir al usuario si quiere ampliar o no el rango de búsqueda.

Además, debe mostrar el tiempo que tarda en recorrer cada uno de los intervalos y encontrar la mochila supercreciente.

Ejemplo

>> cp =[106 265 583 1219 2703 1285 2782 383 1296 3903]

mu =5234

>> cripto_shamir_zippel(cp, mu)

vamos a buscar en el rango [1 , 2048]

(espera respuesta de ordenador)

Elapsed time is 0.019980 seconds.

hemos encontrado la mochila simple

s1 = 2 5 11 23 51 123 250 501 1012 2345

>> cripto_shamir_zippel([471,785 ,1413,3611,8007,19468,15349,5384,18165,2824,13165], 25000)

vamos a buscar en el rango [1 , 4096]

(espera respuesta de ordenador)

Elapsed time is 0.001338 seconds.

no hemos encontrado la mochila, si quieres ampliar el rango, responde 1 1

Ahora nuestro rango será [1, 8192]

(espera respuesta de ordenador)

Elapsed time is 0.001585 seconds.

no hemos encontrado la mochila, si quieres ampliar el rango, responde 1 1

Ahora nuestro rango será [1, 16384]

(espera respuesta de ordenador)

Elapsed time is 0.001862 seconds.

hemos encontrado la mochila simple

s1 = 3 5 9 23 51 124 257 512 2345 5432 12345