Códigos y Criptografía - Curso 2019-2020

Práctica 6: Cifrado RSA. Autentificación de firma.

- Usaremos las funciones de prácticas anteriores **letranumero (texto)** y **potencia (c,d,n)**

NOTA: El texto de los ejemplos no hay que reproducirlo tal cual, lo importante es que los programas/funciones hagan lo que se pide, y que si tienen que solicitar algún dato al usuario sean precisos en la descripción de los valores solicitados.

1.- Hacer un programa, genero_clave.m, para generar las claves públicas y privadas que necesitamos para cifrar con RSA.

El programa debe pedir al usuario los valores de los primos p y q. Una sugerencia, para facilitar la introducción de estos valores, es pedirle al programa que muestre en primer lugar un listado de números primos hasta un cierto valor.

Si los números anteriores son lo suficientemente grandes puede considerar $e = 1 + 2^{2^4} = 65537$, y si no o bien solicitarlo al usuario o generar cualquier valor para *e* válido de la manera que consideréis más oportuna.

Ejemplo

>> genero_clave

Introduzca el valor del primo p: 1187

Introduzca el valor del primo q: 1171

n, que formará parte de las claves es: n=1389977

Estamos buscando e, que formara parte de la clave privada y que debe cumplir gcd(e,fiden)=gcd(e,p·q)=gcd(e,1387620)=1,

hemos seleccionado 65537

Buscamos d de forma que d sea el inverso de e = 65537 modulo fiden=1387620,

hemos seleccionado 924713

La clave privada es (n.d)=(1389977,924713)

La clave publica es (n,e)=(1389977,65537)

2.- Función doble = letra2numeros (texto)

A cada letra del texto, la función le debe asociar su correspondiente valor de Z_{27}, con dos dígitos: *a:00, b:01, …, z:26.*

Entradas: *texto:* el texto llano

Salida: *doble:* cadena numérica formada por los números asociados a cada letra del texto.

Ejemplo

```
>> doble=letra2numeros('hola, vamos a cifrar con RSA')
doble =  '07151100220012151900020805180018021513181900'
```

3.- Función blo = prepa_num_cifrar (tama, bloque)

Función que convierte una cadena numérica en bloques de un tamaño dado, después convierte dichos bloques en números y los almacena en un vector. Si es necesario para completar el último bloque deberemos añadir varios *30's* y/o un *0*.

Entradas:

tama: tamaño de los bloques.

bloque: cadena numérica.

Salida: *blo*: vector formado por los números que se corresponden con cada uno de los bloques.

Ejemplo

```
>> blo = prepa_num_cifrar(7,'83629486523')
blo =   8362948     6523300

>> blo= prepa_num_cifrar(7,'836294806523')
blo =   8362948      652330
```

4.- Función desci = cifro_rsa (e, n, texto)

Función que debe hacer los siguientes pasos: pasar *texto* a una cadena numérica usando 2 dígitos por letra, calcular el tamaño de los bloques a partir de *n*, aplicar la función anterior para obtener los bloques de números, y cifrarlos según el sistema RSA usando la clave pública *(n,e)*.

Entradas:

e y n: clave pública para el cifrado RSA.

texto: texto que queremos cifrar.

Salida: *cifrado*: vector formado por los bloques ya cifrados.

Ejemplo

>> cifrado=cifro_rsa(65537, 2726447, 'cifrando con RSA')

cifrado = 670406 2123352 740929 1523275 1351881

5.- Función cifrado = cifro_rsa_num (e, n, blo)

Función igual a la anterior, salvo que en vez de recibir un texto, recibe ya los bloques de números.

Entradas:

e y n: clave pública para el cifrado RSA.

blo: vector de números.

Salida: *cifrado*: vector formado por los bloques introducidos ya cifrados.

Ejemplo

>> cifrado=cifro_rsa_num(65537, 2726447, [20805 180013 31502 151318 190030])

cifrado = 670406 2123352 740929 1523275 1351881

6.- Función desci = num_descifra (n, bloque_numero)

Función que transforma un vector numérico en letras (dos dígitos por letra). Debe completar los bloques para que tengan longitud *dígitos(n)-1*, concatenarlos y agrupar de dos en dos, eliminar los posibles *30*'s y/o *0* que pueda haber al final, y pasar a letras.

Entradas:

n: número que va a determinar el tamaño de los bloques, *dígitos(n)-1.*

bloque_numero: vector numérico.

Salida: *desci*: cadena alfabética con el texto asociado a los bloques de números.

Ejemplo

```
>> descifro=num_descifra(2127781,[104 201530])

descifro =    'abeto'
```

7.- Función descifro_num =descifro_rsa_num (d, n, cifrado_numero)

Función que aplica la función de descifrado del método RSA a un vector numérico (que supuestamente se habrá cifrado previamente con la clave pública), utilizando la clave privada proporcionada.

Entradas:

(d, n): clave privada.

cifrado_numero: vector numérico, se supone que cifrados según la clave pública con RSA.

Salida: *descifro_num*: vector numérico obtenidos tras aplicar la función de descifrado con RSA.

Ejemplo

```
>> cifrado=[403866  424206  786183  950614  1268222  1245474  747657
1069757]

>> descifro_num=descifro_rsa_num(924713, 1389977, cifrado)

descifro_num =  161518   50813   161503   41215   190304   190208   51800   183030
```

8.- Función descifrado = descifro_rsa (d, n, cifrado_numero)

Función de descifrado según el método RSA, que siga todos los pasos estudiados.

Entradas:

(d, n): clave privada.

cifrado_numero: vector numérico, se supone que cifrados según la clave pública con RSA.

Salida: *descifrado*: texto claro.

Ejemplo

```
>> cifrado=[403866  424206  786183  950614  1268222  1245474  747657
1069757]

>> descifro=descifro_rsa(924713,1389977,cifrado)

descifro =   'porfinpodemosdescifrar'
```

9.- Construye todo lo que necesites para hacer una autentificación de firma mediante el método RSA

Ejemplo

La clave publica de A es (na,ea)=(27371551,13)
La clave privada de A es (na,da)=(27371551,18941533)
La clave publica de B es (nb,eb)=(492859,179)
La clave privada de B es (nb,db)=(492859,422459)

El mensaje que A quiere enviar, junto con su firma es:

mensaje = 'el programa funciona byalma'

Los dos criptogramas que envia A a B son:

cif_mens = 432488 192897 450957 295922 319626 81530 184771 165686 440500 53020

cif_firma_da_eb = 259007 68799 439509 59081

B comienza el descifrado

El mensaje con la firma que recibe es:

mensaje = 'elprogramafuncionabyalma'

B obtiene la firma y comprueba su autentificación:

firma = 'byalma'