

RSA

Con bloques

FUNCIONES DEL RSA

RSA::RSA(int bits)

- Int bits: # de bits, posición del bit prendido
- Bits: Intervalo de números en base decimal que serán usados por el RSA.

| Valor de la posición | $(2^{2048})-1$ | $(2^{1024})-1$ | $(2^{512})-1$ |
|-----------------------|-------------------------------|-------------------------------|-----------------------------|
| #BITS | 2048 | 1024 | 512 |
| Intervalo # decimales | $(2^{2048})/2 - (2^{2048})-1$ | $(2^{1024})/2 - (2^{1024})-1$ | $(2^{512})/2 - (2^{512})-1$ |

- Si RSA(1024) significa que tienen que generar los número primos entre $(2^{1024})/2$ y $(2^{1024})-1$.

RSA::RSA(int bits)

- Int bits: # de bits, posición del bit prendido
- Bits: Intervalo de números en base decimal que serán usados por el RSA.

| Valor de la posición | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|-----------------------|---------|--------|-------|--------------|------|-----|-----|-----|
| # Bits | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Intervalo # decimales | 255-128 | 127-64 | 63-32 | 31-16 | 15-8 | 7-4 | 3-2 | 1-0 |

- Si RSA(5) significa que tienen que generar los número primos entre 31 y 16.

RSA::RSA(int bits)

- Por la criba de Eratóstenes: debería votar el vector de primos

| 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|
| 31 | 29 | 23 | 19 | 17 |

- Entonces los valores de p , q y e serán cualquiera de los valores que esté en el vector de primos.

Class RSA

- class RSA
- {
- public:
- RSA(int bits);
- RSA(ZZ,ZZ);
- virtual ~RSAC(void);
- ZZ exponenciacion(ZZ , ZZ);
- ZZ resto_chino(ZZ);
- string cifrar(string);
- string descifra_mensaje(string);
- long euclides(ZZ a, ZZ b);
- vector extendido_euclides(ZZ a, ZZ b);
- ZZ inversa(ZZ a, ZZ b);
- ZZ generar_aleatorio(int);
- bool test_primalidad(ZZ);
- private:
- ZZ d;
- ZZ e;
- ZZ N;
- ZZ p;
- ZZ q;
- string alfabeto;
- };

Funciones

- **ZZ generar_aleatorio(int bits)**
 - Dado un entero n, esta funcion consigue un numero aleatorio de n bits.
 - Generará aleatorios entre el intervalo_mayor e intervalo_menor.

Funciones

- **ZZ test_de_primalidad(ZZ numero)**
 - Dado un numero n , determinar si es primo o bien es compuesto. Aplicando metodos que permiten establecer la primalidad de un numero sin necesidad de factorizarlo.

Funciones

- **vector criba_Eratóstenes (int intervalo mayor, int intervalo menor);**
 - Por la criba de Eratóstenes: debería votar el vector de primos

| 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|
| 31 | 29 | 23 | 19 | 17 |

- Entonces los valores de p, q y e serán cualquiera de los valores que esté en el vector de primos.

Funciones

- **vector criba_Eratóstenes (int intervalo mayor, int intervalo menor);**
 - Ingreso: intervalo de números. Los números del intervalo van de acuerdo al número de bits que ingresen.
 - Por ejemplo: Si el número de bits es 5. El intervalo mayor es 31 y el menor 16.
 - Retorna el vector de primos.

Funciones

- **vector criba_Eratóstenes (int intervalo_mayor, int intervalo_menor);**

- Deben generar un vector inicial para almacenar los números desde el 31 (intervalo_mayor) al 16 (intervalo_menor).

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

- Eliminar múltiplos de 2 ó generar un vector de impares del 31 al 16.

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 17 | 19 | 21 | 23 | 25 | 27 | 29 | 31 |
|----|----|----|----|----|----|----|----|

- $3^2 = 9$. $9 < 31$. Eliminar múltiplos de 3: cómo no están **2.3**, **3.3**, (menores que 3). Iniciamos desde **3.4** =12 es menor que 17, no se considera, **3.5** =15 es menor que 17, **3.6**=18 es par, **3*7** =21, SE ELIMINA, **3*8**=24, no está, **3*9**=27 SE ELIMINA, **3*10**= 30 es par, **3*11**=33, FUERA DE RANGO

| | | | | | |
|----|----|----|----|----|----|
| 17 | 19 | 23 | 25 | 29 | 31 |
|----|----|----|----|----|----|

- $5^2 = 25$. $25 < 31$. Eliminar múltiplos de 5: cómo no están **2.5**, **3.5**, **4.5** (menores que 5). Iniciamos desde **5.5** =25, SE ELIMINA, **5.6** =30 es par, **3*7**=35, FUERA DE RANGO

| | | | | |
|----|----|----|----|----|
| 17 | 19 | 23 | 29 | 31 |
|----|----|----|----|----|

- $6^2 = 36$. $36 < 31$. No se cumple. Termina la criba. Retorna el vector
- Nota: No es necesario considerar los múltiplos pares.

Funciones

- **ZZ inversa (ZZ e, ZZ N)**

- Recibe el valor de la clave pública y de N.
- Si el $\text{mcd}(e, N) == 1$ (Usar Euclides), e tiene inversa.
 - De lo contrario llamar a la función `generar_primo()`. Hasta que se cumpla la condición anterior.
- Llamar a la función `extendido_euclides(e, N)`. Retorna un vector de 3 números: x, y, d.
 - Ojo: Euclides extendido calcula d nuevamente, pero eso ya fue calculado en el paso anterior. (Estarían redundando). Pueden mejorar el código aquí!!.
- Verificar si x es positivo retornar x, de lo contrario convertir el valor de x a positivo ($x = N + x$).

TAREA 1: GENERAR CLAVES

Paso 1:

- El alfabeto será una cadena de string formado por los caracteres:
abcdefghijklmnopqrstuvwxyz(espacio
blanco)ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
- Cada alumno deberá generar los valores de:
 - p y q generados aleatoriamente.
 - Condición primos: **Criba de Eratóstenes**
 - N **debe ser del tamaño de bits ingresados en el constructor**
 - $N = p \times q$
 - Clave pública “e” generada aleatoriamente
 - Condición $\text{mcd}(e, \phi N) = 1 \dots e = [2, \phi N]$. **Cálculo de un número aleatorio**
 - **Condición $\text{mcd}(e, \phi N) = 1$**
 - Calcular clave privada “d”
 - Para calcular la inversa:
 - **(Usar el Teorema de Euler, Fermat, Factorización)**

Ejemplo: Paso 1

- Beatriz genera los siguientes valores:

$$p=17$$

$$q=59$$

$$N=1003$$

$$\phi N = (17-1) (59-1)$$

$$e=3$$

$$d=619$$

Paso 2

- Los datos serán almacenados en un archivo txt llamado “directorio publico.txt”

| Nombre | N | e | correo |
|-------------|------|-----|--------------------|
| Beatriz G | 1003 | 3 | beatriz@gmail.com |
| Carlos R | 4027 | 5 | carlosr@gmail.com |
| Adriana Paz | 1937 | 229 | adrianag@gmail.com |
| | | | |

- **Nota** valores solo con fines de ejemplificación

TAREA 2: ENVIAR MENSAJE

Paso 3

- Adriana transforma el mensaje en una secuencia de d-dígitos (d=número de dígitos de la letra del alfabeto más significativa). Uds. deben considerar el espacio en blanco

| | | | | | | | | |
|----|----|----|----|----|----|----|----|----|
| c | o | m | e | h | e | r | e | # |
| 02 | 14 | 12 | 04 | 07 | 04 | 17 | 04 | 26 |

- Dividir el mensaje en bloques:
 - El valor de N tiene 4 dígitos:
 - El mensaje es dividido en N-1 bloques, $k=3$

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 021 | 412 | 040 | 704 | 170 | 426 |
|-----|-----|-----|-----|-----|-----|

Paso 4

- Adriana cifra el mensaje, para eso tiene que
 - Leer los valores de “N” y “e” que generó Beatriz del archivo “directorio publico.txt”
 - $N=1003$ y $e=3$
- Aplica la fórmula del cifrado para cada bloque del mensaje
 - $C_1=21^3 \bmod 1003 = 234$
 - $C_2=412^3 \bmod 1003 = 353$
 - $C_3=40^3 \bmod 1003 = 811$
 - $C_4=704^3 \bmod 1003 = 54$
 - $C_5=170^3 \bmod 1003 = 306$
 - $C_6=426^3 \bmod 1003 = 545$
- Cada C_i debe tener el mismo número de dígitos de N
 - 023403530811005403060545

Paso 5

- Adriana envía al correo electrónico de Beatriz el mensaje cifrado más la firma digital en un archivo txt llamado cifrado

023403530811005403060545

cifrado.txt

TAREA 3: RECIBIR MENSAJE

Paso 6

- Beatriz recibe el archivo cifrado.txt, conteniendo el mensaje cifrado.
- El mensaje es dividido en N-dígitos (N tamaño de dígitos de N)
- Beatriz lee cada bloque del mensaje y haciendo uso de su clave privada “d”
- Aplica la fórmula de descifrado:
 - Usar algoritmo exponenciación rápida binaria – Teorema Chino del Resto
 - $D_1 = C^{619} \bmod 1003$

Paso 7

- Divide el texto cifrado en bloques del tamaño de N de Beatriz
 - $0234^{619} \bmod 1003 = 21$
 - $0353^{619} \bmod 1003 = 412$
 - $0811^{619} \bmod 1003 = 40$
 - $0054^{619} \bmod 1003 = 704$
 - $0306^{619} \bmod 1003 = 170$
 - $0545^{619} \bmod 1003 = 426$

Paso 8

- Como el mensaje está formado por bloques de tamaño 3 (N-1), entonces el resultado de Di se rellena con ceros para completar los 3 dígitos:
- 021-412-040-704-170-426-000-705-061-634-031-901-511-430-024-600-372-323
- Reagrupando en 2-dígitos (por el tamaño de la letra del alfabeto más significativa) tenemos:
 - 02 -> c
 - 14->o
 - 12->m
 - 04->e
 - 07->h
 - 04->e
 - 17->r
 - 04->e
 - 26->#

| | | | | | | | | |
|----|----|----|----|----|----|----|----|----|
| 02 | 14 | 12 | 04 | 07 | 04 | 17 | 04 | 26 |
| c | o | m | e | h | e | r | e | # |

Observe que cada bloque de 2 no debe ser mayor que el tamaño del alfabeto