INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECANICA Y ELECTRICA UNIDAD CULHUACAN

MAESTRÍA EN INGENIERÍA EN SEGURIDAD Y TECNOLOGÍAS DE LA INFORMACIÓN

IMPLEMENTACIÓN DEL ALGORITMO DE CIFRADO RSA

Presenta:

Jonathan Eduardo Garciá García

jgarciag1404@alumno.ipn.mx

Profesor:

Dr. Eduardo Vázquez Fernández

Fecha:

06/06/2022

https://jon2g.github.io/WebSharpRSA/

NOTA:

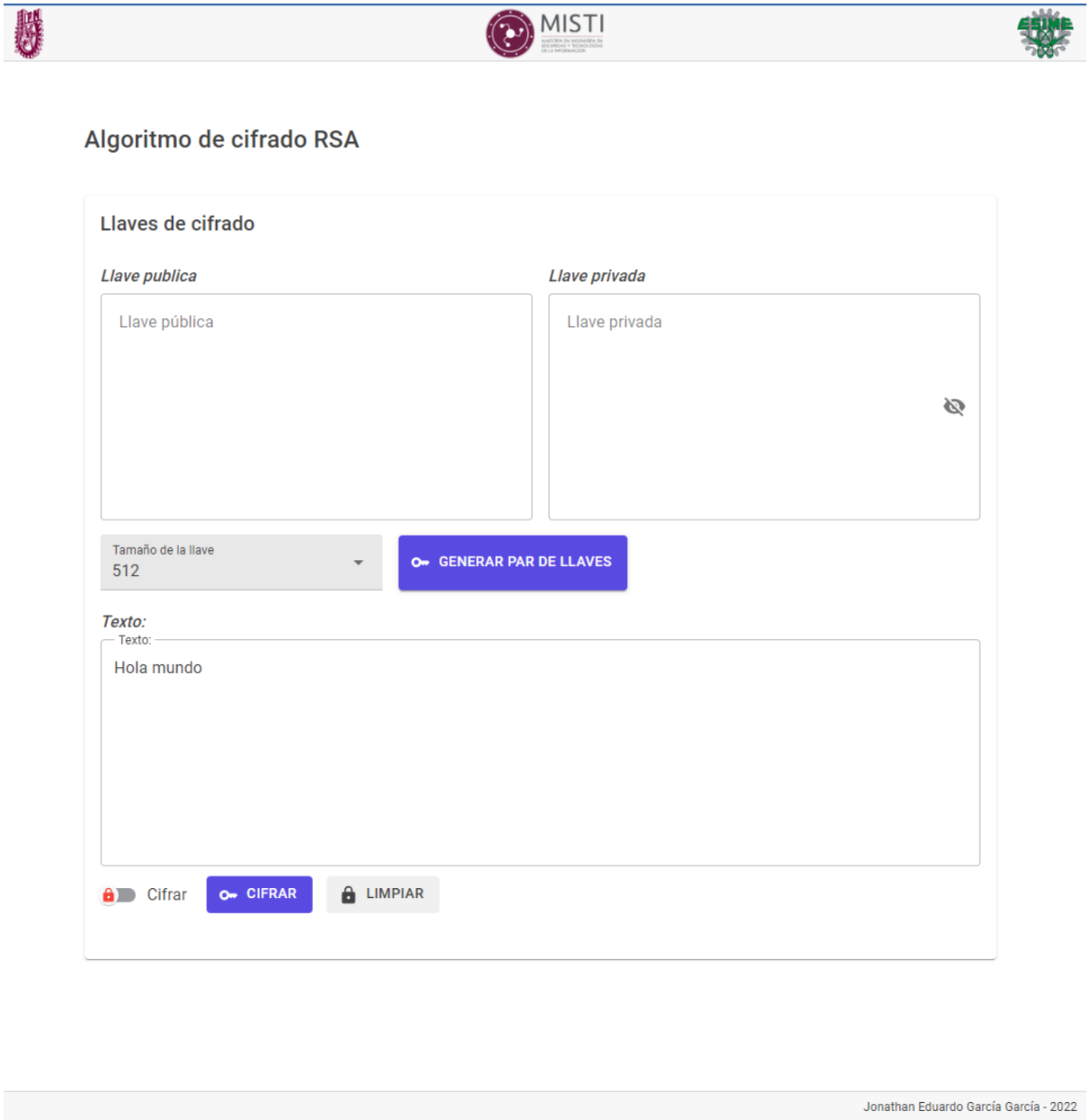
El programa esta publicado en la siguiente dirección:

https://jon2g.github.io/WebSharpRSA/

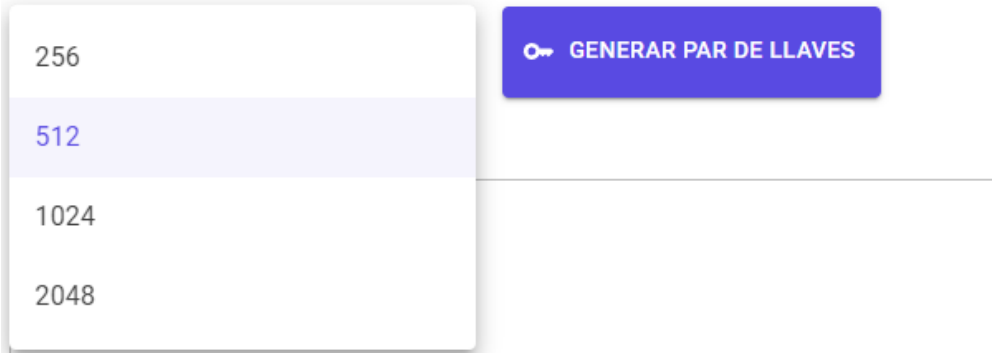
El código fuente puede consultarse en:

https://github.com/Jon2G/WebSharpRSA

Capturas de pantalla del programa en funcionamiento:



1) Procedemos a generar el par de llaves:



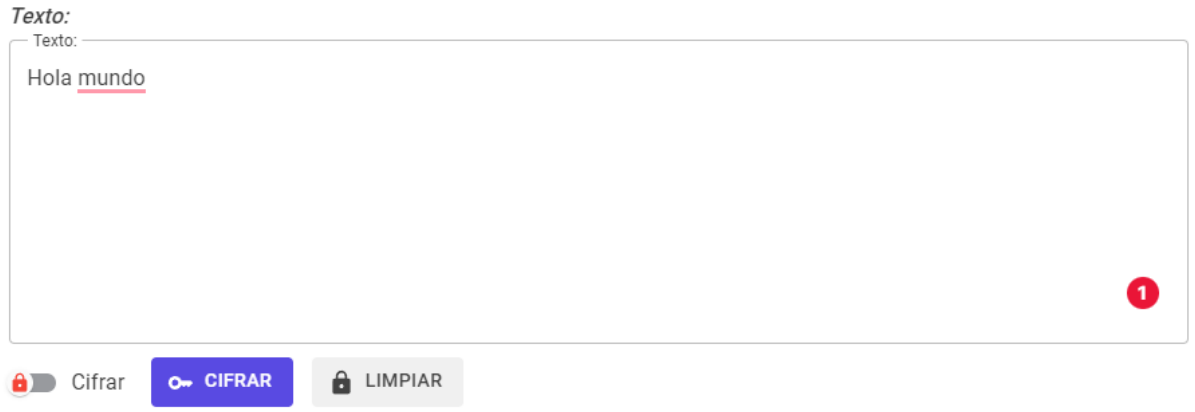
Podemos escoger entre distintos tamaños para n

El par de llaves se muestra en formato Base64 :

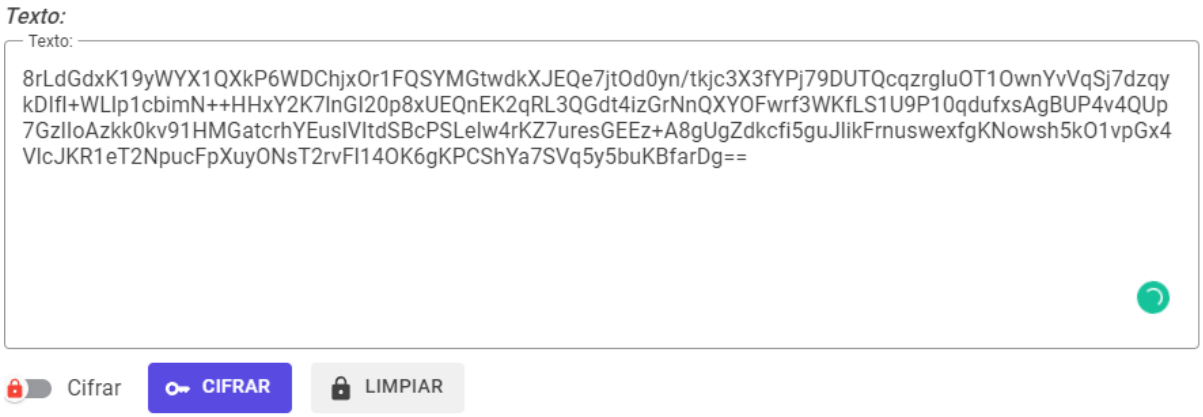
Por defecto la llave privada esta censurada

Cifrado

2) Ingresamos el texto que se desea cifrar en el recuadro



Y presionamos cifrar:



El texto cifrado se imprime en formato Base64

Descifrado

3) Alternamos el estado del switch bajo el texto para cambiar a modo de descifrado

4) Presionamos descifrar (La llave privada debe ser correcta para este paso)

Si todo es correcto se nos mostrará el texto original

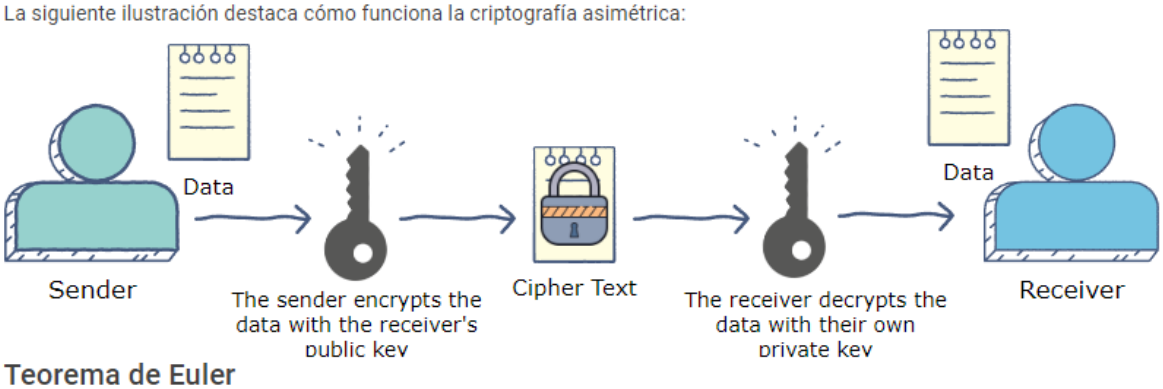
Teoría

El algoritmo de cifrado RSA

El algoritmo RSA es un algoritmo criptográfico asimétrico; esto significa que utiliza una clave pública y una clave privada (es decir, dos claves diferentes vinculadas matemáticamente). Como sugieren sus nombres, una clave pública se comparte públicamente, mientras que una clave privada es secreta y no debe compartirse con nadie.

El algoritmo RSA lleva el nombre de quienes lo inventaron en 1978: Ron Rivest, Adi Shamir y Leonard Adleman.

La siguiente ilustración destaca cómo funciona la criptografía asimétrica:



Teorema de Euler

El teorema de Euler es la base del algoritmo RSA, y es importante entenderlo antes de explicar el algoritmo.
La idea es que si tenemos dos números a y n tal que son coprimos, entonces: $a^{\phi(n)} \equiv 1 \pmod{n}$

Donde ϕn es conocida como la función ϕ de Euler, y para números primos, es: $\phi p = p - 1$
Para un producto de dos números primos pq entonces $\phi(pq) = \phi(p)\phi(q)$

Generando las claves

1. Seleccione dos números primos grandes x,y.
   Los números primos deben ser grandes para que a alguien le resulte difícil descifrar.
2. Calcular $n = x * y$
3. Calcular la función Phi $\phi(n) = (x-1)(y-1)$
4. Seleccionar un número entero e, tal que e sea coprimo con $\phi(n)$ y $1 < e < \phi(n)$
   El par de números (n,e) conforman la llave pública.

   > ***Nota:***
   >
   > Dos enteros son coprimos si solo un el único entero positivo los divide es 1

5. Calcular d de tal forma que $e * d = 1 mod(\phi(n))$

   El par de números n,d conforman la llave privada.

   > d puede ser encontrada utilizando el algoritmo extendido de Euclides

Cifrado

Cada caracter del texto plano debe convertirse en su representación decimal.

Texto	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Valor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

De modo que el texto completo es representado con el número: 12345678910111213141516171819020212223242526

$$C = P^e mod(n)$$

$$C = 12345678910111213141516171819020212223242526^e mod(n)$$

Donde C representa al texto cifrado.

Descifrado

Teniendo el número C que representa al texto cifrado hacemos:

$$P = C^d mod(n)$$

El resultado de P será el número que representa al texto original

Algoritmo de cifrado:

Implementación propuesta:

```
double x = 3;
double y = 7;

double n = x*y;

double e = 2;
double phi = (x-1)*(y-1);
while (e < phi)
{
    if (gcd(e, phi)==1)
    break;
        else
    e++;
}

BigInteger k
BigInteger limit = p * q;
for (BigInteger i = 1; i < limit; i += 2)
{
    if (i != p && i != q){
        if (i>1 && i < phi && phi % i != 0){
            k=i;
            break;
        }
    }
}

BigInteger d = (1 + (k*phi))/e;

BigInteger msg = 20;

// Cifrado
BigInteger c = BigInteger.Pow(msg, e);
c = BigInteger.Mod(c, n);

// Descifrado
BigInteger m =BigInteger.Pow(c, d);
m = BigInteger.Mod(m, n);

//Algoritmo extenido de euclides
int gcd(int a, int h)
{
    int temp;
    while (1)
    {
        temp = a%h;
        if (temp == 0)
          return h;
        a = h;
        h = temp;
    }
};
```