Práctica 4: Cifrado RSA

1. Definir las funciones **primosolostra** y **primoMillerRabin** para los test de Solovay-Strassen y de Miller-Rabin. Dichas funciones deben tomar como variables el rango donde buscar el primo y el número de iteraciones a considerar. En caso de obtener un número que pase los test, debe indicar la probabilidad de que dicho supuesto primo sea un pseudo-Primo. También debe dar como respuesta el tiempo requerido para realizar el test.

2. Definir la función **keygeneration** que toma dos números primos (el programa puede sugerir inicialmente algunos primos para usar, quizás con una tabla) y genera las claves públicas y privadas. Para el valor de e debe dar como opciones: el primo de Fermat $e = 65537$ si los primos son suficientemente grandes, tomar e de forma aleatoria o e dado por el usuario.

3. Reutilizar las funciones que ya tenemos definidas para pasar de texto a cifras, usando la equivalencia del alfabeto con los números de $\mathbb{Z}_{27}$. Utilizar dos cifras para cada letra: $a \to 00$, $b \to 01$, etc. De igual forma, necesitaremos la función que envía cifras a texto.

4. Definir una función **preparenumcipher** que coge una cadena numérica (un texto transformado a su equivalente numérico) y lo divide en bloques de tamaño fijado por n. El programa deberá incluir $30s$ o 0 para rellenar los bloques incompletos. De igual forma, definir una función **preparetextdecipher** que coja un vector numérico, y devuelva una cadena numérica lista para ser traducida a texto.

5. Definir las funciones **rsacipher** y **rsadecipher** que toma bloques numéricos y los cifra de acuerdo a la clave pública $((n, e))$ o la clave privada $((n, d))$ según corresponda.

6. Definir una función **rsaciphertext** y **rsadeciphertext** que haga lo mismo que la anterior, pero iniciando con un texto dado en el caso de la primera, o devolviendo el texto descifrado en el caso de la segunda.

7. Definir adicionalmente **rsaciphertextsign** que permita la autenticación del emisor del mensaje. Esto es, el mensaje debe tomar las claves públicas del receptor (n_B, e_B), las claves privadas del emisor (n_A, d_A), un texto a cifrar *TEXTO* y una firma *FIRMA* y debe generar dos criptogramas C_1 y C_2. C_1 debe ser el cifrado de *TEXTOFIRMA* con la clave pública del receptor mientras que C_2 debe ser el cifrado de *FIRMA* con la clave privada del emisor.

8. Definir **rsadeciphertextsign** que tome dos criptogramas C_1 y C_2 y que devuelva el mensaje descifrado junto con la verificación del emisor (puede simplemente indicarse que el mensaje está autenticado de acuerdo a la clave pública del emisor).

9. Siguiendo el esquema anterior, introducir las funciones necesarias para el cifrado y descifrado de ElGamal.

Recordad que todo el código realizado deberá estar convenientemente documentado.