**Programa de Encriptación de Mensajes Criptográficos**

**Introducción**

El Programa de Encriptación de Mensajes Criptográficos es una herramienta basada en Python diseñada para encriptar y desencriptar mensajes usando la operación XOR. Este programa proporciona una forma simple pero efectiva de asegurar información sensible a través de técnicas criptográficas.

**Instrucciones de Uso**

**Instalación**

1. Asegúrate de tener Python instalado en tu sistema.
2. Instala las dependencias necesarias usando pip install sympy.

**Ejecución del Programa**

1. Abre una terminal o símbolo del sistema.
2. Navega al directorio que contiene los archivos del programa.
3. Ejecuta el programa usando el comando python main.py.
4. Sigue las instrucciones en pantalla para encriptar o desencriptar mensajes.

**Requisitos de Entrada**

* **Mensaje**: Ingresa el mensaje que deseas encriptar o desencriptar.
* **Semilla**: Proporciona un valor de semilla para la generación de números aleatorios (opcional).
* **Tipo de Mensaje**: Elige el tipo de mensaje (por ejemplo, Primer Contacto, Mensaje Regular).

**Formato de Salida**

* Mensaje Encriptado: El resultado de encriptar el mensaje de entrada.
* Clave: La clave criptográfica utilizada para la encriptación.

**Dependencias**

Este programa depende de las siguientes bibliotecas:

* [SymPy](https://www.sympy.org/): Utilizada para generar números primos.

**Algoritmos Utilizados**

**Encriptación XOR**

El programa emplea la operación XOR (OR Exclusivo) para encriptar y desencriptar. Esta operación a nivel de bits combina el mensaje con una clave para producir el mensaje encriptado, y viceversa para la desencriptación.

**Generación de Claves**

Las claves se generan usando números aleatorios y números primos grandes dentro de un rango especificado. La clave generada se utiliza en la operación XOR para encriptar y desencriptar.

**Consideraciones de Seguridad**

* **Fortaleza de la Clave**: Es crucial utilizar claves fuertes para asegurar la seguridad de los mensajes encriptados. Evita usar claves predecibles o débiles.
* **Semilla Segura**: Proporcionar una semilla segura mejora la aleatoriedad de la generación de claves, haciéndola más resistente a ataques criptográficos.
* **Elección del Algoritmo**: Si bien la encriptación XOR es adecuada para propósitos de demostración, considera utilizar algoritmos de encriptación más robustos para datos sensibles en aplicaciones del mundo real.

**Referencias**

* Documentación de la Biblioteca SymPy: https://www.sympy.org/doc

**Información de Licencia**

Este programa se distribuye bajo la Licencia MIT, otorgando a los usuarios la libertad de usar, modificar y distribuir el software dentro de los términos especificados.

**Información de Contacto**

Para preguntas, comentarios o soporte, por favor contacta a Juan Fonseca en [juanjguandique@gmail.com](mailto:juanjguandique@gmail.com) o a Javier Hernández [javihernandez0707@gmail.com](mailto:javihernandez0707@gmail.com)  
Visita el repositorio de GitHub para contribuciones y actualizaciones: <https://github.com/JuanJFG/DSS101_Cryptography>