2024

“Implementación de algoritmo de cifrado polimórfico”

EXPERIMENTO DE INVESTIGACIÓN

JAVIER ENRIQUE HERNÁNDEZ MÁRQUEZ HM210444
JUAN JOSÉ FONSECA GUANDIQUE FG203030

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

Objetivo

- Implementación del algoritmo de cifrado de 64 bits.
- Evaluación de la calidad de la generación de claves.
- Selección y documentación de las funciones de cifrado.
- Prueba de cifrado/descifrado de mensajes.
- Prueba del funcionamiento de los tipos de mensajes intercambiados.

Descripción de las funciones del algoritmo.

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.

encryption_module.py 1 | message_types.py | main.py

src > encryption_module.py > ...

```python
import random
import string
from sympy import primerange

def generate_large_prime():
    """Genera un número primo grande dentro de un rango especificado para la generación de claves."""
    primes = list(primerange(1000000, 2000000))
    return random.choice(primes)

def generate_key(seed, prime, size=64):
    """
    Genera una clave criptográfica de 64 bits utilizando una semilla y un número primo.
    Args:
    Semilla: La semilla utilizada para el generador de números aleatorios.
    primo: Un número primo grande para el cálculo de la clave.
    tamaño: El tamaño en bits de la clave, por defecto es 64 bits.
    Devuelve:
    Un entero que representa la clave criptográfica.
    """
    random.seed(seed)
    key = random.getrandbits(size)
    key = (key * prime) % (1 << size)  # Apply modular arithmetic to keep key within range
    return key

def encrypt_message(message, key):
    """
    Cifra un mensaje utilizando una simple operación XOR con fines de demostración.
    Args:
    Mensaje: El mensaje a cifrar (cadena).
    clave: La clave criptográfica (entero).
    Devuelve:
    El mensaje cifrado como una cadena que contiene todos los caracteres ASCII imprimibles.
    """
    key %= 256  # Ensure key is within range (0-255) for XOR operation
    valid_chars = string.printable
    encrypted_message = ''.join(char for char in ''.join(chr(ord(char) ^ key) for char in message) if char in valid_chars)
    return encrypted_message

    return encrypted_message

def decrypt_message(encrypted_message, key):
    """
    Descifra un mensaje mediante una simple operación XOR.
    Args:
    mensaje_encriptado: El mensaje cifrado (cadena).
    clave: La clave criptográfica utilizada para el cifrado (entero).
    Devuelve:
    El mensaje descifrado como cadena.
    """
    key %= 256  # Ensure key is within range (0-255) for XOR operation
    decrypted_message = ''.join(chr(ord(char) ^ key) for char in encrypted_message)
    return decrypted_message

```

Documentación del código fuente.

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.

Conclusiones

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

Bran, Flores, Hernández (2019), Cryptography model to secure IoT device

endpoints, based on polymorphic cipher OTP, IEEE Press.

Información de Licencia

Información de Contacto

Para preguntas, comentarios o soporte, por favor contacta a Juan Fonseca en juanjguandique@gmail.com o a Javier Hernández javihernandez0707@gmail.com

Visita el repositorio de GitHub para contribuciones y actualizaciones:

https://github.com/JuanJFG/DSS101_Cryptography