CRIPTOGRAFÍA CON PYTHON

Criptografía Clásica o Tradicional

Autor: Enrique Melchor Iborra Sanjaime (em.iborrasanjaime@edu.gva.es)

Contenido

1. Objetivos

- Comprender qué es la criptografía y por qué es importante.
- Conocer los diferentes tipos de criptografía y sus usos.
- Introducir los conceptos matemáticos básicos necesarios para entender los algoritmos criptográficos.
- Conocer la historia de la criptografía clásicos
- Comprender las técnicas básicas de cifrado

---

2. Historia y conceptos básicos

¿Qué es la criptografía?

La criptografía es el estudio y la práctica de técnicas para asegurar la comunicación frente a adversarios. Se utiliza para proteger datos sensibles, garantizar la privacidad y validar la autenticidad.

Tipos de criptografía

- **Criptografía clásica o pre-era-Digital**:
  - Cifrados por sustitución
    - César, Vigenère, sustitución, espejo, polibio, ...
  - Cifrados por transposición
- **Esteganografía**
  - Antigua
  - En la era Digital
- **Criptografía simétrica**:
  - Usa la misma clave para cifrar y descifrar.
  - Ejemplo: Algoritmo AES.
  - **Ventaja**: Es rápido.
  - **Desventaja**: Requiere un intercambio seguro de claves.
- **Funciones hash**:
  - Convierte datos en una "huella digital" única.
  - No es reversible (no se puede "descifrar").
  - Ejemplo: SHA-256.
  - **Uso común: verificar la integridad de los datos.**
- **Criptografía asimétrica**:
  - Usa un par de claves (clave pública y clave privada).
  - Ejemplo: RSA.
  - **Ventaja**: La clave pública se puede compartir libremente.
  - **Desventaja**: Es más lento que la criptografía simétrica.
- **Criptografía post-cuántica**

Principios fundamentales

1. **Confidencialidad**: Garantiza que solo los destinatarios autorizados puedan acceder a la información.
2. **Integridad**: Asegura que los datos no hayan sido alterados durante la transmisión.
3. **Autenticación**: Verifica la identidad de las partes involucradas en la comunicación.
4. **No repudio**: Impide que una parte niegue haber realizado una acción específica.

---

3. Matemáticas básicas para criptografía

3.1. Álgebra modular

El álgebra modular es esencial en la criptografía, especialmente en sistemas como RSA.

Definición: Dados dos números aa y nn, a mod n es el residuo al dividir a entre n.

- Ejemplo en python : 17 % 5 Resultat: 2

Propiedades importantes:

1. $(a+b) \bmod n = [(a \bmod n)+(b \bmod n)] \bmod n$
2. $(a \cdot b) \bmod n = [(a \bmod n) \cdot (b \bmod n)] \bmod n$

3.2. Generación de números primos

Los números primos son fundamentales para algoritmos asimétricos como RSA.

- **Definición**: Un número es primo si solo es divisible por 1 y por sí mismo.
- Algoritmos comunes para comprobar primalidad:
  - Prueba de divisores.
  - Algoritmo de prueba de primalidad de Fermat.

Ejemplo en Python:

```
def es_primo(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# Uso:
print(es_primo(17))  # True
print(es_primo(18))  # False
```

Explicación del Algoritmo

1. **Casos especiales:**
   - Si $n \leq 1$, no es primo.
2. **Reducir iteraciones con raíz cuadrada:**
   - No necesitas verificar divisores mayores que sqrt{n}, porque si $n=a \times b$, entonces al menos uno de los factores (a o b) debe ser $\leq$ sqrt{n} .

Ejemplo optimizado

```
def es_primo(n):
    """
    Determina si un número es primo.

    Parámetros:
    - n: Número entero a verificar.
```

```
    Retorna:
    - True si n es primo.
    - False si n no es primo.
    """
    if n <= 1:
        return False  # Los números menores o iguales a 1 no son primos
    if n <= 3:
        return True  # 2 y 3 son primos
    if n % 2 == 0 or n % 3 == 0:
        return False  # Descarta múltiplos de 2 y 3

    # Verificar divisores desde 5 hasta la raíz cuadrada de n
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False  # n no es primo si es divisible por i o i + 2
        i += 6  # Incremento optimizado (prueba solo números de la forma 6k ± 1)

    return True  # n es primo si no se encontraron divisores
```

Explicación del Algoritmo

- Los números 2 y 3 son primos, así que se manejan como casos directos.
- Los múltiplos de 2 y 3 se descartan rápidamente.
- Todos los números primos mayores que 3 son de la forma 6k±1. Por lo tanto, una vez descartados los múltiplos de 2 y 3, puedes probar solo los números de esa forma.

3.3. Claves públicas y privadas

1. **Claves públicas**: Usadas para cifrar o verificar.
2. **Claves privadas**: Usadas para descifrar o firmar.
3. **Relación matemática**: En RSA, las claves se generan usando factorización de números grandes.

---

4. Definición de criptografía clásica

La **criptografía clásica** es un término utilizado para describir los métodos de cifrado que se desarrollaron y usaron antes de la era de las computadoras modernas. En general, estos métodos se basan en transformaciones manuales o mecánicas para cifrar y descifrar mensajes

Denominación y características principales

1. **Definición**:
   Se refiere a los sistemas de cifrado que no dependen de algoritmos computacionales complejos, sino de operaciones relativamente simples, como sustituciones y transposiciones.
2. **Otros nombres comunes**:
   - **Criptografía tradicional**.

- **Criptografía histórica**.
- **Criptografía manual** (por la falta de computadoras para su implementación).

3. **Características principales**:
    - Basada en algoritmos simples que pueden implementarse manualmente.
    - Usa alfabetos y sustituciones como componentes clave.
    - No es resistente frente a ataques modernos debido a su simplicidad.
    - Generalmente simétrica, lo que significa que el mismo algoritmo y clave se usan tanto para cifrar como para descifrar.

Ejemplos de criptografía clásica

1. **Cifrados por sustitución**:
   Cambian cada letra o símbolo del texto original (texto plano) por otro. Ejemplos:
    - **Cifrado César**: Desplaza cada letra un número fijo de posiciones en el alfabeto.
    - **Cifrado de Vigenère**: Usa una clave para aplicar múltiples desplazamientos, haciéndolo más seguro que el César.
    - **Alfabeto desordenado**
    - **Cifrado por pares**
    - **Cifrado de polibio**

2. **Cifrados por transposición**:
   Reorganizan las letras del texto plano según una regla específica. Ejemplos:
    - **Cifrado por columnas**: Escribe el mensaje en una cuadrícula y reorganiza las columnas para cifrarlo.
    - **Cifrado escítala**: Utilizado en la antigua Esparta, empleaba un bastón para transponer letras.
3. **Sistemas mecánicos**:
   Aunque no totalmente manuales, algunas herramientas, como la **máquina Enigma**, son consideradas dentro del ámbito clásico, ya que anteceden a la criptografía moderna digital.

Diferencia con la criptografía moderna

- La criptografía clásica se basa en métodos simples y claves cortas, mientras que la moderna utiliza algoritmos complejos y claves largas generadas computacionalmente.
- La criptografía moderna también incluye cifrados asimétricos (como RSA), que no existían en la criptografía clásica.

5. Cifrados por Sustitución

Los **cifrados por sustitución** son un tipo de criptografía clásica en los que cada símbolo o letra del texto original (**texto plano**) se reemplaza por otro símbolo o letra siguiendo una regla específica. Este tipo de cifrado es uno de los métodos más antiguos de ocultar mensajes y ha sido utilizado en diferentes formas a lo largo de la historia.

Características principales

1. **Sustitución uno a uno**:
   Cada símbolo del texto plano se sustituye por otro, generalmente basado en una clave.
2. **Determinístico**:
   El mismo símbolo del texto plano siempre se transforma en el mismo símbolo del texto cifrado, a menos que se use un sistema más avanzado como el cifrado polialfabético.
3. **Simplicidad**:
   Fácil de implementar manualmente, pero también vulnerable a ataques con herramientas simples, como el análisis de frecuencia.

Tipos de Cifrados por Sustitución

Los cifrados por sustitución pueden dividirse en **monoalfabéticos** y **polialfabéticos**, según el número de alfabetos usados para la sustitución.

2.1. Cifrados Monoalfabéticos

En estos cifrados, se usa un solo alfabeto de sustitución para todo el texto. Son los más simples, pero también los más fáciles de romper.

Ejemplos:

- **Cifrado César**:
- En el Cifrado César, cada letra en el mensaje original se reemplaza por otra letra que está a un número fijo de posiciones hacia adelante o hacia atrás en el alfabeto. Este número es conocido como **desplazamiento** (también llamado "clave").
  - Ejemplo con un desplazamiento de 3:
    Texto plano: `HOLA`
    Texto cifrado: `KROD`

    Funcionamiento básico
    Imagina un alfabeto estándar (sin letras con acentos ni caracteres especiales):
    `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`
    Por ejemplo, si usamos un desplazamiento de 3 (clave = 3), el alfabeto se vería así:

```
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
|
V
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

    Entonces, con un desplazamiento de 3, la letra **A** se convierte en **D**, **B** en **E**, **C** en **F**, y así sucesivamente.

- Fórmula matemática:
  C=(P+k) mod n , donde P es el valor de la letra en el texto plano, k es el desplazamiento, y n es el tamaño del alfabeto.

- **Cifrado por Alfabeto Desordenado**:
  - Usa un alfabeto reordenado como clave.
    - Ejemplo: (este en concreto se denomina ATBASH o ESPEJO)
      Clave: `ZYXWVUTSRQPONMLKJIHGFEDCBA`
      Texto plano: `HOLA`
      Texto cifrado: `SLOR`

- **Cifrado de sustitucion por Pares:**
  - En este cifrado, cada par de letras es sustituido por otro par de letras de acuerdo con una clave. Es una variación de los cifrados monoalfabéticos clásicos, donde en vez de cifrar letra por letra, se cifran combinaciones de letras
    - Claves de sustitución:
      Clave: `(AB) → XY, (CD) → ZW, (EF) → UV, (GH) → PQ, etc.`
      Texto plano: `HOLA`
      Texto cifrado: `JYRC`

- **Cifrado de polibio**
  - El **Cifrado de Polibio** es un cifrado de sustitución que utiliza una **tabla de 5x5** con las letras del alfabeto para codificar y decodificar los mensajes. Es una variante de cifrado de sustitución que, en lugar de sustituir una letra por otra, sustituye un par de letras (coordenadas) que corresponden a la posición de la letra en una tabla.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

  - **"HOLA"** se cifra como: **23 34 31 11**

2.2. Cifrados Polialfabéticos

En estos cifrados, se usan múltiples alfabetos de sustitución, alternando entre ellos según una regla (normalmente, basada en una clave). Esto hace que un mismo símbolo del texto plano pueda convertirse en diferentes símbolos en el texto cifrado.

Ejemplo:

- **Cifrado de Vigenère**:
  Usa una palabra clave para seleccionar diferentes alfabetos de sustitución.
  - Ejemplo con clave `KEY`:
    Texto plano: `ATAQUE`
    Texto cifrado:

    - A con K = K
    - T con E = X
    - A con Y = Y
    - ... Resultado: `KXYQEI`
- **Cifrado Beaufort**:
  Similar al cifrado de Vigenère, pero utiliza un enfoque matemático diferente para seleccionar los alfabetos.

Fortalezas y Debilidades

Ventajas:

- Simples de entender e implementar.
- Adecuados para ocultar mensajes en tiempos antiguos.

Desventajas:

- **Vulnerables al análisis de frecuencia**:
  En idiomas como el español, ciertas letras son más comunes (`E`, `A`, `O`, etc.). Esto permite descifrar cifrados monoalfabéticos fácilmente.
- **No resisten ataques modernos**:
  Herramientas computacionales pueden romper incluso cifrados polialfabéticos básicos.

Ataques contra los Cifrados por Sustitución

1. **Análisis de Frecuencia**:
   Identifica las letras más comunes en el texto cifrado y las compara con las frecuencias de letras típicas en el idioma del mensaje.
2. **Ataques por Texto Plano Conocido**:
   Si un atacante conoce partes del texto plano y el texto cifrado, puede inferir la clave.
3. **Fuerza Bruta**:
   En cifrados simples como César, probar todas las posibles claves es rápido y sencillo.

Usos Históricos

- **Cifrado César**: Usado por Julio César para comunicarse con sus generales.
- **Cifrado de Vigenère**: Llamado el "cifrado indescifrable" en su época, aunque fue roto posteriormente.

---

6. Cifrados por Transposición

Los **cifrados por transposición** son un tipo de técnica de cifrado utilizada en la criptografía clásica. En este método, las letras del texto original (texto plano) no se sustituyen por otras letras, como en los cifrados por sustitución, sino que se **reorganizan** según un patrón o regla específica. El objetivo es cambiar el orden de los caracteres para dificultar la comprensión del mensaje.

Características principales

1. **Reorganización de caracteres**:
   El orden de las letras del texto plano se altera, pero las letras mismas permanecen sin cambio.
2. **Claves o patrones definidos**:
   La reorganización depende de una clave, que determina cómo se distribuyen los caracteres.
3. **Simetría**:
   Los cifrados por transposición son simétricos, lo que significa que el emisor y el receptor usan la misma clave para cifrar y descifrar.
4. **Seguridad relativa**:
   Aunque más seguros que algunos cifrados simples por sustitución, los cifrados por transposición pueden ser vulnerables a análisis de patrones si no se aplican correctamente.

Tipos de cifrados por transposición

1. **Transposición por columnas**: Este es uno de los métodos más comunes. El texto plano se escribe en una cuadrícula de filas y columnas, y luego se reordena tomando las columnas en un orden específico.

    - **Cifrado**:

        1. Escribe el texto plano en una cuadrícula con un número fijo de columnas.
        2. Reorganiza las columnas según una clave.
        3. Lee las letras columna por columna según el orden definido por la clave.

    - **Ejemplo**: Texto plano: `DEFENDTHEBASE`
      Clave: `3 1 4 2`
      Esquema de la cuadrícula (4 columnas):

```
D E F E
N D T H
E B A S
E
```

Orden de lectura (clave): `3 1 4 2`
Texto cifrado: `FNEDEETDHBAS`

    - **Descifrado**:

        1. Construye una cuadrícula con las columnas reordenadas según la clave.
        2. Reconstruye el mensaje original leyendo fila por fila.

---

2. **Cifrado de ruta**: En este método, las letras del texto plano se colocan en una cuadrícula, pero se leen siguiendo una ruta específica (zigzag, espiral, diagonal, etc.).
    - **Cifrado**:
        1. Escribe el texto plano en una cuadrícula de tamaño definido.
        2. Define una ruta (por ejemplo, leer en zigzag o en espiral).
        3. Lee las letras según esa ruta.
    - **Ejemplo**: Texto plano: `ATTACKATDAWN`
      Ruta en espiral:

```
A T T A
C K A T
D A W N
```

Ruta de lectura en espiral: `ATTACKAWNDT`

- **Descifrado**: Coloca las letras cifradas siguiendo la misma ruta y reconstruye el texto plano.

---

3. **Cifrado escítala**: Utilizado por los espartanos en la antigüedad, este método empleaba un dispositivo físico: un bastón (escítala) alrededor del cual se enrollaba una tira de pergamino. Las letras se escribían a lo largo de las filas creadas por la tira.
    - **Cifrado**:
        1. Enrolla la tira de pergamino en el bastón.
        2. Escribe el texto a lo largo del bastón.
        3. Al desenrollar, las letras quedan aparentemente desordenadas.
    - **Descifrado**: Desenrolla la tira en un bastón del mismo diámetro para leer el mensaje.

Fortalezas y debilidades

Fortalezas:

- Fácil de implementar manualmente.
- Puede combinarse con otros métodos (como sustitución) para mejorar la seguridad.
- Adecuado para entornos de recursos limitados donde no se dispone de tecnologías modernas.

Debilidades:

- **Análisis de patrones**: Si el atacante conoce el idioma del mensaje, puede identificar patrones y deducir la clave.
- **Longitud fija**: Los mensajes cortos son más vulnerables, ya que la estructura puede ser deducida fácilmente.
- **Vulnerabilidad ante permutaciones**: Los métodos modernos permiten probar automáticamente muchas combinaciones posibles.

---

7. Combinaciones con otros cifrados

Los cifrados por transposición a menudo se combinaban con métodos de sustitución para aumentar la seguridad. Por ejemplo:

1. **Cifrado de Vigenère combinado con transposición**:
    - Primero se sustituye el texto con Vigenère.
    - Luego se reordena usando un método de transposición.
2. **Cifrado por bloques modernos**:
   Los algoritmos modernos como AES se basan en principios similares, aplicando transposiciones junto con sustituciones en bloques de datos.

Aplicaciones actuales

Aunque los cifrados por transposición no se usan directamente en la criptografía moderna debido a sus vulnerabilidades, el concepto sigue presente en:

- **Algoritmos de cifrado simétrico**: Como parte de técnicas de permutación.
- **Esteganografía**: Donde se oculta información reorganizando datos en estructuras específicas.

---

8. Actividades

Generación de números primos

- Implementa una función para encontrar todos los números primos en un rango.

```
def primos_en_rango(inicio, fin):
    return [x for x in range(inicio, fin + 1) if es_primo(x)]

print(primos_en_rango(10, 50))
```

- Calcula la cantidad de números primos en un rango grande (por ejemplo, hasta 10,000).
- Implementa funciones para sumar, restar y multiplicar números bajo un módulo dado. (los resultados no pueden sobrepasar el módulo)
- Crea una función para cifrar un texto sumando un valor constante a cada carácter (Cifrado César). Funcines `chr y ord`
- Crea una función que dado un array o string , cree otro con un desplazamiento N y modulo la longitud del primer array/string

Cifrado y Descifrado César

Objetivo

Crear un programa que permita:

1. Cifrar un mensaje ingresado por el usuario usando una clave numérica.

2. Descifrar el mensaje cifrado usando la misma clave.
3. Validar que el descifrado devuelve el mensaje original.

Actividades Relacionadas

1. **Experimentar con diferentes claves**:
    - Prueba claves negativas o mayores que 26. Explica por qué funcionan igualmente.
2. **Desafío: Romper un cifrado sin clave**:
    - Implementa un programa que descifre un mensaje probando todas las claves posibles (ataque de fuerza bruta).
3. Guardar mensajes cifrados en un archivo de texto
4. Cargar mensajes desde un archivo de texto
5. Soporte para cifrados avanzados, como el cifrado Vigenère.
6. Gestión de claves diferentes por mensaje (almacenando la clave junto al mensaje en el archivo).
7. Agregar validaciones de la clave
8. Cambiar el formato de archivo a JSON
9. Programa con Interfaz Gráfica (GUI)
    - Crea una versión con interfaz gráfica utilizando:
    - `tkinter`: Para una solución básica y rápida.
    - `PyQt` o `Kivy`: Para una interfaz más compleja.

Actividades Extra Relacionadas con el Programa

1. **Formato JSON para Mensajes y Claves**
    - Cambiar el formato de almacenamiento a un archivo JSON, organizando los datos como:

```
[
  {"mensaje": "Mtqf Rzspi", "clave": 5},
  {"mensaje": "Krod Pxqgr", "clave": 3}
]
```

    - Usa el módulo `json` para leer y escribir mensajes.
2. **Validación de Entrada del Usuario**
    - Implementa controles para:
        - Evitar claves negativas o no numéricas.

- Asegurarte de que el mensaje no esté vacío.

3. **Historial de Mensajes Descifrados**
   - Guarda los mensajes descifrados en una lista temporal mientras el programa está en ejecución.
   - Permite mostrar un historial de todos los mensajes descifrados en la sesión.
4. **Exportación de Mensajes Descifrados**
   - Añade la opción de exportar mensajes descifrados a un archivo separado para referencia futura.
5. **Soporte Multilingüe**
   - Adapta el programa para trabajar con alfabetos adicionales, como el cirílico o caracteres especiales en español (ñ, tildes, etc.).
   - Esto implicaría usar mapas personalizados en lugar de dependencias ASCII directas.

Actividades Avanzadas

6. **Cifrado con Caracteres Unicode**
   - Expande el cifrado para incluir caracteres Unicode (por ejemplo, emoticonos o caracteres chinos).
   - Usa el rango completo de Unicode en lugar de limitarte a letras.
7. **Simulación de Ataques de Fuerza Bruta**
   - Implementa un ataque para descifrar mensajes cifrados probando todas las claves posibles.
   - Muestra cada resultado y permite al usuario elegir el correcto.
8. **Interfaz Gráfica (GUI)**
   - Crea una versión con interfaz gráfica utilizando:
     - `tkinter`: Para una solución básica y rápida.
     - `PyQt` o `Kivy`: Para una interfaz más compleja.
9. **Autenticación de Usuario**
   - Agrega una capa de autenticación básica con un nombre de usuario y contraseña para proteger los mensajes cifrados.
   - Usa hashing para almacenar las contraseñas (por ejemplo, `bcrypt` o `hashlib`).
10. **Soporte para Claves Dinámicas**
    - En lugar de una clave fija, permite al usuario usar una clave basada en un patrón, como la fecha actual, un PIN, etc.

Cifrado Avanzado

11. **Cifrado Vigenère**

- Implementa el cifrado Vigenère, que utiliza una palabra clave en lugar de un número para cifrar el mensaje.
- Ejemplo: Usar la palabra `CLAVE` para cifrar "HOLA" generará un patrón dinámico de desplazamientos.

12. **Generación de Claves Aleatorias**

- Genera claves aleatorias para cada mensaje y almacénalas junto con el mensaje cifrado.

13. **Cifrado Reversible con XOR**

- Implementa un cifrado basado en XOR (puerta lógica) que permita cifrar y descifrar usando la misma clave.

14. **Simulación de un Cifrado por Bloques**

- Divide el mensaje en bloques de tamaño fijo y aplica el Cifrado César o un cifrado más complejo por bloques.

Colaboración y Seguridad

15. **Envío Seguro de Mensajes**

- Simula el intercambio de mensajes cifrados entre dos usuarios con claves precompartidas.
- Implementa un intercambio básico de claves simulando un esquema de criptografía simétrica.

16. **Protección del Archivo**

- Cifra el archivo donde se almacenan los mensajes usando una clave maestra.

Optimización y Escalabilidad

19. **Uso de Bases Diferentes**

- Permite al usuario elegir diferentes bases para las claves (binaria, hexadecimal, etc.).

20. **Desempeño con Grandes Cantidades de Datos**

- Cifra y descifra archivos grandes o múltiples mensajes en un solo lote.

21. **Comparación de Algoritmos**

- Implementa varios cifrados básicos (César, Vigenère, XOR) y mide cuál es más eficiente con diferentes tipos de datos.

Proyectos Relacionados

22. **Juego de Criptografía**

- Crea un juego interactivo donde los usuarios resuelvan acertijos basados en mensajes cifrados.

23. **API de Cifrado**

- Desarrolla una API sencilla con Flask o FastAPI que permita enviar un mensaje, cifrarlo y devolver el resultado.

24. **Aplicación Móvil**

- Transforma el programa en una app móvil usando herramientas como `Kivy`.

25. **Generador de Contraseñas Seguras**

- Usa conceptos de criptografía para crear contraseñas seguras basadas en patrones definidos por el usuario.