

Sistema de Cifrado de Archivos con Interfaz Gráfica en Python

David Ricardo Ordoñez Mora

Estudiante Bootcamp de Ciberseguridad - TalentoTech

Funcionamiento / Explicación: <https://youtu.be/2BscvpKvjsY>

Sistema de Cifrado y Descifrado de Archivos .txt, .pdf y .docx utilizando Algoritmos Vigenère y Fernet con Interfaz Gráfica en Python

Objetivos

Objetivo General

Desarrollar un sistema de cifrado y descifrado de documentos que soporte los formatos .txt, .pdf y .docx, utilizando los algoritmos de Vigenère y Fernet, con interfaz de usuario gráfica y registro de errores.

Objetivos Específicos

- Implementar el algoritmo clásico de Vigenère para cifrar y descifrar texto alfabético.
- Aplicar el cifrado simétrico Fernet para garantizar confidencialidad en archivos.
- Leer y escribir archivos en formatos .txt, .pdf y .docx.
- Desarrollar una interfaz gráfica con Tkinter para facilitar el uso del sistema.
- Registrar errores del sistema en un archivo de log para su depuración.

Investigación de Herramientas para Cifrado de Archivos y Carpetas

¿Qué es el cifrado?

El **cifrado** es una técnica que transforma un mensaje para que solo lo puedan leer personas autorizadas. Imagina que escribes una carta y la encierras en una caja con candado: Solo quien tenga la llave puede leerla. Así funciona el cifrado: **protege la información**, haciendo que un archivo se vea ilegible si alguien sin permiso lo abre.

“Piensa en el cifrado como poner tu información en una caja fuerte. Solo quien tenga la **clave correcta** podrá abrirla” Es una de las herramientas principales de la **ciberseguridad**, usada para proteger:

- Mensajes personales
- Archivos confidenciales
- Documentos en la nube
- Contraseñas y más

Cifrado de Vigenere

Es un método de cifrado por sustitución poli alfabética. Usa una palabra clave para determinar cómo cambiar cada letra del texto original.

- Ejemplo: Si la clave es SOL y el texto es HOLA, cada letra del texto se desplaza en el alfabeto según la letra correspondiente de la clave.

¿Cómo lo implementa Python?

En tu proyecto, el cifrado Vigenère:

- Recorre cada letra del texto.
- Usa funciones como `ord()` y `chr()` para convertir letras en números y viceversa.
- Aplica una fórmula matemática que toma en cuenta la clave y la posición de cada letra.

Este método funciona con archivos .txt, .pdf y .docx. Python extrae el texto del archivo, lo cifra letra por letra, y luego lo vuelve a guardar.

¿Para qué lo usamos en el proyecto?

- Para demostrar cómo se construyen los algoritmos de cifrado desde cero.
- Para enseñar a los usuarios cómo una clave afecta el contenido del mensaje.
- Para fines didácticos, mostrando que incluso métodos antiguos pueden proteger información básica.
-

Cifrado de Fernet

Fernet es parte del módulo `cryptography` de Python y representa un algoritmo de cifrado simétrico seguro. Usa una clave secreta aleatoria para cifrar y descifrar datos.

“Es como tener una llave maestra que convierte archivos legibles en datos cifrados imposibles de leer sin la clave”.

¿Cómo lo implementa Python?

En tu proyecto:

- Se genera una clave Fernet con `Fernet.generate_key()` y se guarda.
- El archivo se lee como texto.
- El texto se convierte a bytes (`.encode('utf-8')`).
- Se cifra con `fernet.encrypt()`.

- Se guarda como un archivo .cif (binario cifrado).
- Luego se puede descifrar con `fernet.decrypt()` y reconstruir el archivo en su formato original (.txt, .pdf, .docx).

“Python garantiza que cada dato cifrado tenga integridad y autenticidad, protegiéndolo incluso de alteraciones”.

¿Para qué lo usamos en el proyecto?

- Para ofrecer una alternativa **segura y realista** de cifrado de archivos personales.
- Para practicar el uso de librerías profesionales de criptografía en Python.
- Para enseñar a proteger información de forma **automatizada** y reutilizable, incluso para quienes no saben programar.

¿Por qué usar ambos métodos en un mismo proyecto?

Método	Propósito en el proyecto
Vigenère	Enseñar la lógica básica del cifrado con letras y claves.
Fernet	Mostrar un ejemplo de cifrado seguro y aplicable en la vida real .

- Con esto, el usuario puede **ver la evolución del cifrado**: desde lo clásico y manual hasta lo moderno y automatizado.

¿Cómo facilita Python este proyecto?

- Permite leer y escribir distintos tipos de archivo fácilmente (txt, pdf, docx).
- Usa módulos como `cryptography`, `docx`, `reportlab`, y `PyPDF2`.
- Maneja errores con logging y archivos .log para registrar fallos.
- Ofrece una interfaz gráfica (tkinter) que **no requiere conocimientos de programación** para usarla.

Objetivo de la Investigación

Analizar y justificar el uso de herramientas de cifrado de archivos para carpetas locales, enfocadas en criptografía simétrica realizable en Python, con énfasis en su implementación práctica.

Herramientas Investigadas

1. **Fernet** (biblioteca **cryptography**) - Fernet es un sistema de cifrado simétrico que utiliza AES-128-CBC + HMAC-SHA-256, con claves de 32 bytes, IV aleatorio y verificación de integridad.
 - a. **Ventajas:**
 - i. - Asegura confidencialidad, integridad y autenticidad con una API sencilla.
 - ii. - Ideal para cifrar bytes de archivos completos.
 - b. **Justificación:** Combina robustez (AES+CBC+HMAC) con facilidad de uso en Python, apropiado para proteger información local en contextos académicos y profesionales.
2. **PBKDF2** (derivación de clave por contraseña) - **PBKDF2** es una función estándar (RFC 2898, PKCS #5 v2.0) que transforma contraseña + sal + múltiples iteraciones en una clave simétrica segura.
 - a. **Ventajas:**
 - i. - Añade una capa de seguridad al convertir contraseñas en claves robustas.
 - ii. - Ideal para usar con Fernet en escenarios donde la clave deriva de una contraseña de usuario.
 - b. **Justificación:** Fortalece sistemas basados en password, evitando ataques de fuerza bruta.
 - c. **Video:** <https://www.youtube.com/watch?v=fKw4L-mmV0c> PBKDF2 tutorial in Python – Python Basics
3. **Tkinter** - GUI estándar de Python, multiplataforma, ideal para prototipos rápidos.
 - a. **Usos en proyecto:**
 - i. - Ventana principal
 - ii. Selección de archivos (filedialog)
 - iii. Entradas de clave y botones de acción
 - iv. Mensajes emergentes con messagebox
 - b. **Justificación:** No requiere dependencia externa, permite construir interfaces funcionales sin complicaciones.
 - c. **Video:** https://www.youtube.com/watch?v=ouO_QRJtvNE - Open Files Dialog Box – Python Tkinter GUI Tutorial #15
4. **PyInstaller** Genera ejecutables (e.g. .exe) de scripts Python.
 - a. **Ventajas:**
 - i. Facilita distribución sin exigir Python en el sistema.
 - ii. Compatible con GUIs como Tkinter.
 - b. **Justificación:** Ideal para entregar aplicaciones a usuarios finales sin entornos de desarrollo.

5. **logging (Python estándar)** - Biblioteca para registrar eventos: errores, advertencias, información.
 - a. **Implementación:** Errores durante cifrado/descifrado se guardan en logs/error.log
 - b. **Justificación:** Mejora trazabilidad de errores y soporte técnico, manteniendo calidad y profesionalismo en el sistema.

Comparativa de herramientas

Herramienta	Cifrado seguro	Interfaz	Portable	Requiere instalación	Programable
Python + Fernet	✓ AES-128/CBC	✓ GUI	✓ .exe	✗ Python o exe	✓
VeraCrypt	✓ Muy fuerte	✓ GUI	✗	✓ Instalación	✗
7-Zip + AES	✓ Medio	✗ CLI	✓	✓ Instalación	✗
WinRAR con clave	✗ Menos fuerte	✓ GUI	✓	✓ Instalación	✗

Conclusión

La solución combinada:

1. **Fernet + PBKDF2:** cifrado fuerte y seguridad de clave
2. **Tkinter:** interfaz accesible sin dependencias externas
3. **PyInstaller:** ejecutable para usuarios finales
4. **logging:** calidad profesional y seguimiento de errores es una plataforma robusta, portable y adecuada para uso académico, personal y empresarial.

Tecnologías utilizadas

1. **Lenguaje** - Python
2. **Entorno desarrollo** - VisualStudio Code
3. **Herramientas** - Github
4. **Librerías:**
 - a. “**tkinter**” - Interfaz gráfica
 - b. “**cryptography**” - Cifrado Fernet
 - c. “**PyPDF2**” - Lectura de archivos PDF
 - d. “**python-docx**” - Lectura y escritura de archivos “.docx”
 - e. “**reportlab**” - Escritura de archivos PDF
 - f. “**logging**” - Registro de errores

Características

1. Interfaz gráfica con botones para:
2. Seleccionar archivos
3. Ingresar clave para Vigenère
4. Cifrar/descifrar usando Vigenère
5. Cifrar/descifrar usando Fernet
6. Soporte para múltiples tipos de documentos: “.txt”, “.pdf”, “.docx”
7. Registro automático de errores en “logs/error.log”
8. Archivos generados automáticamente con sufijos descriptivos:
 - a. “_Vigenere”, “_Vigenere_Descifrado”
 - b. “_Fernet”, “_Fernet_Descifrado”
9. Paleta de colores personalizada para la interfaz.

Requisitos para ejecutar

Instala las dependencias con el siguiente comando:

- pip install cryptography python-docx PyPDF2 reportlab

Estructura del Proyecto

```

|—— __pycache__/
|—— claves/
|   |—— clave.key      # Clave de Fernet generada
|—— funciones/
|   |—— __init__.py
|   |—— fernet.py      # Lógica Fernet.
|   |—— vigenere.py    # Lógica Vigenere.
|—— logs/
|   |—— error.log      # Registro de errores.
|—— interfaz.py        # Interfaz gráfica con Tkinter.
|—— main.py            # Ejecuta cifrado y descifrado con Vigenère.
|—— mainFernet.py     # Ejecuta cifrado y descifrado con Fernet.
|—— prueba_gui.py     # Prueba interfaz gráfica.
|—— requerimientos.txt # Dependencias.
|—— readme.md         #

```

Descripción de Archivos y Módulos

funciones/vigenere.py

Contiene funciones para:

- leerArchivo (): lee archivos .txt, .pdf y .docx.
- guardarArchivo (): guarda texto en dichos formatos (PDF usa ReportLab).

- `cifrarVigenere ()`: implementa el algoritmo clásico de Vigenère.
- `descifrarVigenere ()`: realiza el descifrado correspondiente.

funciones/fernet.py

Contiene funciones para:

- `generarClave ()`: genera y guarda una clave Fernet.
- `cargarClave ()`: lee una clave Fernet desde un archivo.
- `leerContenido ()`: lee el contenido según el tipo de archivo.
- `escribirContenido ()`: guarda el contenido descifrado en su formato original.
- `cifrarArchivo ()`: cifra un archivo con Fernet.
- `descifrarArchivo ()`: descifra y reconstruye el archivo.

interfaz.py

- Interfaz gráfica desarrollada con Tkinter.
- Permite seleccionar archivo, algoritmo y tipo de operación.
- Ejecuta procesos y muestra resultados.

main.py

- Ejecuta directamente el cifrado y descifrado con Vigenère sobre un archivo.

mainFernet.py

- Ejecuta directamente el cifrado y descifrado con Fernet sobre un archivo.

logs/error.log

- Registro automático de errores y excepciones para seguimiento de fallos.

Requisitos

Incluidos en requerimientos.txt:

- PyPDF2
- reportlab
- python-docx
- tk
- cryptography

Instalación:

- `pip install -r requerimientos.txt`

Ejecución - Vía consola:

- `python main.py`
- `python mainFernet.py`

Vía interfaz gráfica:

- `python prueba_gui.py`

Notas Adicionales

- El sistema guarda por defecto los archivos cifrados y descifrados en la misma ubicación del archivo original, con sufijos `_Vigenere`, `_Vigenere_Descifrado`, `_Fernet`, `_Descifrado`, según corresponda.
- Para cifrado Fernet, el archivo `claves/clave.key` debe existir o ser generado antes.