PRÁCTICA 01

TEORÍA DE CÓDIGOS Y CRIPTOGRAFÍA

Aplicación Criptográfica a Desarrollar

Autores
Antonio Miguel Almagro Valles
Javier Diaz Vílchez
Adrián Jiménez Benítez
David Montoya Segura

Asignatura
Teoría de Códigos y Criptografía

Titulación
Grado en Ingeniería Informática

Contenido

1. Enunciado.

Esta primera tarea requiere redactar un breve informe (archivo pdf) que describa la aplicación criptográfica que desarrollará el grupo de trabajo a lo largo del curso. La aplicación debe cifrar y descifrar información, que puede ser un único archivo o una carpeta elegida por el propio usuario, un grupo de archivos o carpetas, o todo el disco duro.

La descripción debe incluir su funcionamiento así como la interacción con el usuario, que podrá ser activa, es decir, el usuario tiene que ejecutarlo, por ejemplo, una aplicación para cifrar fotografías personales; o pasivo, es decir, la aplicación se ejecutará como un script sin la intervención del usuario, como podría ser el caso de un ransomware.

En el último caso, debemos proporcionar una forma de recuperar la información cifrada como se requiere anteriormente. Otros aspectos que se deben dar son el Sistema Operativo seleccionado, la plataforma de desarrollo, el lenguaje de programación así como las bibliotecas necesarias, los archivos de destino (archivos que serán cifrados/descifrados) y cualquier otra información que los autores consideren relevante.

Es recomendable un (primer) diagrama de bloques para comprender mejor el funcionamiento.

2. Introducción.

El desarrollo de la aplicación se llevará a cabo en Python, aprovechando su amplia gama de librerías enfocadas en criptografía. A lo largo de este informe, se analizarán y evaluarán diferentes librerías disponibles para Python, con el fin de identificar la más adecuada para cumplir con los requisitos de seguridad y eficiencia del proyecto. La elección de la mejor librería se va a fundamentar en los factores como rendimiento, facilidad de uso y compatibilidad con las operaciones necesarias.

3. Descripción de las librerías.

pyAesCrypt:

Es una librería de Python que implementa cifrado simétrico mediante AES256 en modo CBC. Permite cifrar y descifrar archivos y flujos binarios, compatible con el formato de AES Crypt. Se puede ajustar el tamaño del búfer para optimizar el rendimiento y soporta encriptación en memoria. Sin embargo, no incluye soporte para cifrado de clave pública, firma digital, compartición de secretos como Shamir, ni algoritmos como El Gamal. Es una herramienta eficaz para proteger archivos con cifrado simétrico, pero no aborda otros métodos criptográficos avanzados.

Fernet:

Forma parte del paquete Cryptography y permite cifrar y descifrar datos de forma simétrica, donde se utiliza una misma clave, garantizando privacidad y autenticidad, ya que incluye un mecanismo de verificación de integridad en sus operaciones, generando una firma con HMAC, que es un método criptográfico que combina una función hash como SHA-256 con una clave secreta. Cualquier cambio en los datos indicaría que se han manipulado o corrompido, y esto se puede saber porque el código de autentificación no coincidirá al verificar su integridad.

Además, permite la rotación de claves y soporte para caducidad de estas, lo que significa que se puede gestionar el ciclo de vida de las claves. Como ventajas, es una librería sencilla de utilizar y proporciona una seguridad robusta, ya antes mencionada, sin embargo, no está preparada para poder encriptar archivos de gran tamaño, ya que todo debe cargarse en memoria, por lo que el trabajo ya se limitaría a archivos pequeños. Pese a ello, tampoco cuenta con muchas funcionalidades, ya que solo utiliza cifrado simétrico (AES), y lo que se busca para este proyecto es, además de poder encriptar, repartir la clave secreta a varios usuarios.

pycrypto:

Es el predecesor de PyCryptodome, puesto que su desarrollo se detuvo hace tiempo. teniendo su última versión en el año 2013. Se trata de un conjunto de herramientas criptográficas que permite implementar y utilizar diferentes algoritmos de cifrado, entre los cuales destacamos algoritmos de cifrado simétrico (AES y DES, entre otros) que requieren mantener una clave compartida entre emisor y receptor, algoritmos de cifrado asimétrico (RSA y DSA) empleando claves públicas y privadas, hashing y firmas digitales (SHA y MD5) y generación de números aleatorios seguros para generar claves y tokens de seguridad. Como se ha mencionado anteriormente, se trata de una librería desactualizada cuya última versión es de hace unos años, además de que no presenta una gran variedad de funcionalidades y, aunque es eficiente, concluímos en que no es la mejor opción de entre las propuestas para este proyecto.

Cryptography:

Dicha librería se considera la “biblioteca estándar de criptografía” para Python, ofreciendo interfaces tanto de alto nivel como de bajo nivel, las características principales de dicha librería son la siguiente: Cifrado simétrico (Fernet), explicado anteriormente, cifrado asimétrico (soporta algoritmo como RSA y DSA para cifrado y firma digital), implementa algoritmos de hash seguros como SHA-256, soporta funciones como PBKDF2 HMAC para derivar claves seguras a partir de contraseñas.
Como se puede observar es una librería muy completa, pero tiene algunas desventajas y limitaciones como, por ejemplo, el rendimiento en lenguajes de bajo nivel, que puede verse afectado de manera negativa o puede tener problemas de compatibilidad con ciertos sistemas operativos y contener algunos errores y vulnerabilidades.
Aún así, la librería que se ha escogido parece adecuarse con las necesidades del proyecto, ya que no tiene dependencias externas significativas, lo que aporta facilidad a la hora de su instalación y uso, también cabe destacar que, para nuevos proyectos se recomienda cryptography por su API moderna y enfoque en la seguridad, aunque no será un problema para la realización del proyecto.

PyCryptodome:

A la hora de elegir una librería criptográfica para el proyecto, PyCryptodome destaca significativamente en comparación con las alternativas evaluadas (PyAesCrypt, Fernet y Cryptography). Esta decisión está respaldada por su vasta gama de funcionalidades, su versatilidad y su compatibilidad con los requisitos específicos del proyecto.

Primero, PyCryptodome incluye cifrado simétrico, soportando algoritmos como AES en varios modos, lo que cumple con la necesidad de seguridad en las comunicaciones. Además, no se limita únicamente a cifrado simétrico como Fernet, sino que también proporciona cifrado asimétrico, incorporando algoritmos como RSA y ElGamal, algo esencial para los escenarios donde se requiere intercambio de claves seguras y autenticación mediante firma digital.

Otra ventaja clave de PyCryptodome es su soporte para la compartición de secretos utilizando el esquema de Shamir, lo cual es un requisito crítico en este proyecto para dividir y repartir claves entre varios usuarios de manera segura. Este es un aspecto que PyAesCrypt y Fernet no soportan, limitando su aplicabilidad para proyectos que requieren criptografía avanzada más allá del simple cifrado de archivos o datos pequeños.

Por otro lado, aunque Cryptography también ofrece una amplia gama de funcionalidades, incluyendo cifrado simétrico, asimétrico y firma digital, PyCryptodome presenta una ventaja adicional en términos de rendimiento y flexibilidad. No solo implementa una mayor variedad de algoritmos de cifrado, sino que también proporciona herramientas criptográficas adicionales como la función RSA-PSS para firmas seguras y la generación de claves criptográficas con mayor control. Además, no presenta los mismos problemas de compatibilidad que Cryptography puede tener en algunos sistemas operativos.

En resumen, PyCryptodome es la opción más adecuada porque no solo cumple con todos los requerimientos del proyecto (cifrado simétrico y asimétrico, firma digital, compartición de secretos, algoritmos como RSA, ElGamal y AES), sino que también ofrece un amplio rango de características adicionales que lo hacen más versátil y robusto frente a las demás opciones.

4. Tabla de comparativa de funcionalidades.

	PyCrypto	Fernet	pyAesCrypt	Cryptography	pyCryptodome
AES	SI	SI	SI	SI	SI
RSA	SI	NO	NO	SI	SI
ELGAMAL	SI	NO	NO	NO	SI
CRYSTAL KYBER	NO	NO	NO	NO	NO
DELLYTIUM	NO	NO	NO	NO	NO
McEliece	NO	NO	NO	NO	NO
Bike	NO	NO	NO	NO	NO
Cifrado simétrico	SI	SI	SI	SI	SI
Cifrado de clave pública	SI	NO	NO	SI	SI
Firma digital	NO	NO	NO	SI	SI
Shamir (compartición de secretos)	NO	NO	NO	SI	SI

Tal y como se puede observar, la librería pyCryptodome es la librería que más funcionalidades de las que se van a utilizar a lo largo del proyecto se dispone. Por lo que la librería con la que se va a trabajar a lo largo del curso y del proyecto de Teoría de Códigos y Criptografía va a ser pyCryptodome.

5. Información del proyecto.

Esta aplicación de cifrado y descifrado será desarrollada en **Python** utilizando la librería **PyCryptodome**, diseñada para ejecutarse en **Windows 11**. El usuario podrá seleccionar de manera interactiva un archivo o carpeta que desea cifrar o descifrar. El cifrado se realizará mediante el algoritmo **AES** en modo CBC, generando una clave que el usuario deberá guardar para realizar el proceso de descifrado posteriormente. La aplicación proporcionará una interfaz sencilla en **Jupyter Notebook** que guiará al usuario paso a paso durante la operación.

El flujo de trabajo de la aplicación comenzará con la selección del archivo o carpeta, seguido del proceso de cifrado o descifrado según la opción elegida por el usuario. En el caso del cifrado, el archivo resultante será encriptado y seguro, mientras que, para el descifrado, se solicitará la clave generada previamente para restaurar el archivo o carpeta original. La aplicación tiene como objetivo ofrecer una herramienta segura y fácil de usar para proteger información confidencial del usuario mediante un enfoque interactivo y sencillo, sin requerir intervención avanzada en el proceso técnico.

Actualmente, con la información disponible, se está desarrollando un ejemplo sencillo de cifrado y descifrado debido a la falta de un conocimiento más profundo en **Teoría de Códigos y Criptografía**. Aunque esta aplicación cumple con los requisitos básicos de seguridad de archivos, es solo un punto de partida. A medida que se avance en la asignatura y se adquieran mayores conocimientos teóricos y prácticos, será posible añadir más complejidad y robustez a la aplicación, integrando algoritmos más avanzados y mejorando aspectos clave de la seguridad, como la gestión de claves y los modos de cifrado.

6. Bibliografía.

[1] Cryptography - enlace (03-10-2024)

[2] pyAesCrypt - enlace (03-10-2024)

[3] Fernet - enlace (03-10-2024)

[4] pyCrypto - enlace (03-10-2024)

[5] pyCryptodome - enlace (03-10-2024)

[6] Jupyter lab - enlace (04-10-2024)