Programa encriptación

Eloy Rico Payá 2ºDAW

Índice

Contenido del entregable

El entregable, además de este documento, contiene los archivos de distribución necesarios para ejecutar el programa con normalidad tanto en Linux como Windows, además de un enlace al repositorio en caso de querer revisar el código fuente (https://github.com/ystrtnkl/EjercicioEncriptacion)

Para abrirlo en Windows: ejecutar encriptacion.exe

Para abrirlo en Linux: ejecutar encriptacion

Además vienen unos archivos llamados: admin.key, admin-privada.pem, admin-publica.pem y claves.bin, todos ellos referentes a como el administrador gestiona el programa (ver guía de usuario)

Manual de usuario

El programa funciona mediante la consola de comandos, aunque es muy interactivo, al abrirlo ofrecerá 4 opciones:

1) Encriptar archivo

2) Desencriptar archivo

3) Generar par de claves RSA

4) Administración (ver claves)

5) Información

6) Salir

1) Encriptar archivo

Permite elegir uno o varios archivos (separados por nombres, se admiten rutas relativas y absolutas) de cualquier tipo (esto incluye multimedia, binarios, ascii…)

En caso de ser más de un archivo, preguntará si se quiere empaquetar dicho archivo usando el estándar zipfile-deflated.

Preguntará si la clave AES128 aleatoria que se generará para la encriptación se quiere mostrar por pantalla o guardar en un archivo a elegir por el usuario.

También da la opción de encriptar el archivo con la clave AES128 mediante RSA, para eso pedirá introducir un nombre y buscará la clave pública referente a ese nombre (consultar opción 3).

Por último, en caso de no haber elegido empaquetar los archivos, da la opción de agregar un sufijo personalizado a los archivos encriptados (.aes por defecto) para distinguirlos de los originales.

Finalmente quedarían los archivos encriptados (o el archivo empaquetado) y la clave AES128 (encriptada o no).

Para más información sobre la encriptación consultar el apartado referente a esto en este documento (este programa admite encriptación simétrica, asimétrica e híbrida).

2) Desencriptar archivo

Para la correcta desencriptación se pedirán distintos datos:

Nombre del archivo empaquetado (en blanco si no se empaquetó, en ese caso se pide posteriormente los archivos separados por espacios a desencriptar)

Nombre bajo el cual se encriptó el archivo con la clave AES128 (en blanco si no se encriptó) (buscará <nombre>-privada.pem en esa ubicación)

Ubicación del archivo con la clave (por cuestiones de seguridad, no admite pegar la clave directamente)

Sufijo para los archivos desencriptados (por defecto .unaes) y así poder diferenciarlos de los encriptados (normalmente .aes)

3) Generar par de claves RSA

Genera un par de claves privada y pública RSA bajo el nombre que se escoja (que no sea admin), de esta manera en la misma ubicación que el ejecutable, se generarán dos archivos: <nombre>-privada.pem y <nombre>-publica.pem, preparadas para ser usadas en el programa e indicadas solo mediante el nombre (también es posible generar un par de claves mediante un programa externo, como gnupg. Solo que luego habría que darles el nombre y ubicación oportunos).

4) Administración (ver claves)

Este programa tiene una funcionalidad de administración. Cada vez que se encripta un archivo se añade la clave AES128 sin encriptar (entre otros datos) en claves.bin, el cual está encriptado bajo la clave en admin.key (AES128) la cual está encriptada mediante RSA usando admin-privada.pem y admin-publica.pem.

Cuando se accede a esta opción, el programa comprobará si tiene acceso de lectura y escritura a estos archivos, y si es así permitirá exportar el contenido de claves.bin sin encriptar a un archivo a elegir por el administrador, de esta manera dicho administrador puede ver todas las claves generadas y desencriptar todos los archivos jamás encriptados en este programa.

Como esta funcionalidad debería estar solo disponible para el administrador, dicha persona tendrá que proteger todos los archivos con admin en su nombre bajo los permisos de administrador del sistema operativo (root en Linux y admin en Windows), de esta manera si el programa se abre sin estos permisos no podrá acceder a estos archivos y por lo tanto no dejará exportar todas las claves.

Con el entregable del programa ya vienen una clave AES128 generada además de unas claves RSA ya hechas para el administrador, para mayor seguridad tendrá que generar claves RSA y una AES128 (se puede hacer directamente desde este programa) para asegurar el archivo **claves.bin**

Los pasos que tendría que seguir el administrador serían estos:

1) Generar una clave AES128 y un par de claves RSA bajo cualquier nombre y reemplazarlas por los archivos **admin.key**, **admin-publica.pem** y **admin-privada.pem**

2) Proteger dichos archivos ya sea mediante el sistema de archivos del sistema operativo o simplemente no dejando a los usuarios acceder mediante admin o hacer nada con los archivos de admin

3) Cuando el administrador quiera ver todas las claves generadas, usará esta opción y exportará una copia legible en un archivo. Lo que pase luego con esa copia es responsabilidad del administrador

5) Información

Indica donde encontrar información relevante (básicamente este documento)

6) Salir

Simplemente cierra el programa de manera más elegante (sin hacer Ctrl + C)

Documentación sobre la implementación

Para crear este programa se usó Python3 (el código se puede ver en el repositorio) junto a las siguientes librerías de uso libre:

cryptography: Para la encriptación y desencriptación, tanto con RSA como con AES (librería externa)

pyinstaller: Para generar un ejecutable de la aplicación, tanto para Linux como para Windows (librería externa)

os: Para la gestión de archivos (lectura y escritura, así como comprobar rutas)

sys: Para funcionalidad variada, como cerrar el programa

shlex: Para un mejor manejo de rutas (ya que el formato de las rutas de los archivos varía entre Linux y Windows, especialmente con rutas absolutas)

zipfile: Para empaquetar y desempaquetar los archivos (funcionalidad disponible a la hora de encriptar varios archivos, usó zipfile-deflated)

Además, se usó Docker para poder gestionar un entorno virtual de Python, pudiendo instalar todas las librerías Pip rápidamente en cualquier ordenador (tanto el Dockerfile como los comandos relacionados están en el repositorio)

Y evidentemente se usaron los estándares de encriptación de RSA (clave pública y privada) y AES128 (criptografía simétrica), los algoritmos necesarios para esto están en la librería cryptography.hazmat

El programa en sí consta de 6 archivos de Python:

encriptacion.py: Archivo principal en el cual se encuentran las funciones que muestra el programa, destacan proceso_encriptar() y proceso_desencriptar() que llaman a las funciones del resto del programa y mantienen la conversación con el usuario

empaquetar.py: Tiene las funciones empaquetar(archivo_zip, archivos, silencioso=False) y desempaquetar(archivo_zip, carpeta = "", silencioso = False). La primera recibe una ruta donde guardar un archivo empaquetado y un array con las rutas de los archivos a empaquetar. La segunda recibe una ruta donde se encuentra el archivo empaquetado y la ruta donde se dejarán los archivos de dentro.

rutas.py: Tiene la función normalizar_ruta(ruta) para estandarizar las rutas relativas y absolutas ya sean de Linux o de Windows

codificarAes.py: Tiene la función generar_clave() (que simplemente genera una clave AES128 aleatoria), encriptar(archivo_in, archivo_out, clave, silencioso = False) que encripta archivo x guardándolo en archivo y bajo la clave especificada y desencriptar(archivo_in, archivo_out, claves, silencioso = False) que hace lo contrario, recibe la ruta de un archivo para guardarlo desencriptado en otro sitio bajo una clave especificada.

clavesRsa.py: tiene la función generar(nombre) que genera un par de claves RSA guardándolas en archivos <nombre>-privada.pem y <nombre>-publica.pem, así como las funciones encriptar(nombre, texto) y desencriptar(nombre, texto) para hacer dicha acción sobre un texto buscando la clave con ese nombre. (no se pueden generar claves bajo el nombre de admin)

administracion.py: Contiene funciones relacionadas con la administración del archivo con la copia de todas las claves (claves.bin): autenticar() comprueba que el programa tenga permisos sobre los archivos de admin, guardar_registro_clave(clave, nombre, archivos, sufijo) se ejecuta cada vez que se encripta un archivo, guardando en claves.bin la información y la clave (desencriptándolo para escribir en el y luego volviendo a encriptarlo con clave.key y admin-publica.pem). Y la función solo accesible por el administrador mostrar_claves(archivo_out) que guarda una versión en formato texto y sin encriptar del archivo claves.bin (ver guía de usuario para saber que tiene que hacer el administrador)

Ciberseguridad y algoritmos

Aquí un poco de información sobre los dos métodos de encriptación usados en el programa

Por RSA

Las claves RSA son el método de encriptación asimétrica por excelencia, que se asimétrica quiere decir que se encripta con una clave pero se desencripta con otra totalmente distinta.

Pongamos que persona A quiere enviar información (no muy grande ya que está pensado para textos y archivos pequeños) a persona B, asegurándose de que solo persona B pueda leerla. Podrían utilizar un sistema de encriptación simétrica, de tal manera que ambas personas conozcan la contraseña y puedan encriptar/desencriptar esa información, pero esto sería un problema ya que antes persona A le tendría que mandar la contraseña a persona B

Lo que propone RSA es lo siguiente: persona B genera un par de claves, una privada y una pública. Para esto se usan unos algoritmos matemáticos que hacen que ambas claves estén “enlazadas”, y lo que se encripte con una solo se desencripte con la otra.

Por lo tanto persona B le compartiría su clave pública a persona A (aquí no hay ningún peligro, para lo único que sirve la clave pública es para encriptar), entonces persona A encriptaría la información mediante su clave pública y le mandaría la información encriptada a persona B.

Después de esto, solo persona B puede desencriptar dicha información ya que solo persona B tiene la clave privada asignada a dicha clave pública, por lo tanto, a no ser que se la roben, no habrá nadie más capaz de desencriptar esa información.

Este método se usa en este programa para encriptar claves del algoritmo visto en el siguiente apartado, el usuario introduce su clave pública para encriptar y cuando quiera desencriptar introduciría su clave privada (este segundo paso tiene que ser con extremada precaución, por ejemplo escondiendo la clave privada)

Por AES128

La encriptación mediante RSA está pensada para información no tan grande, como archivos pequeños o textos (no archivos de varios Gb como pueden ser archivos multimedia), si se intenta hacer eso el proceso tardará mucho y quedará un archivo mucho más pesado.

Por eso en este programa se usa AES128, que es un método de encriptar de manera simétrica (misma clave para encriptar y desencriptar) preparado para volúmenes de información mucho más grandes.

Primero de genera una clave aleatoria de 128 bits (la cual luego sí se encripta mediante RSA) y luego se encripta el archivo/archivos (en caso de ser varios se empaquetan en uno solo). De esta manera el archivo(s) resultantes son ilegibles a no ser que se tenga la clave antes generada, la cual además se había encriptado mediante RSA, por lo tanto este programa tiene un método de seguridad mediante criptografía e híbrida (simétrica y asimétrica).