Confidencialidad de la información

Contenido

Que algoritmos de cifrado son actualmente los mejores/más seguros?

Uno de los más usados y el que pondría primero es el algoritmo AES, en específico **AES-256-CBC** es un algoritmo de cifrado simétrico que utiliza una clave de 256 bits, funciona en modo de encadenamiento de bloques cifrados (CBC) y se utiliza ampliamente para proteger datos.

Es un algoritmo de cifrado simétrico, lo que significa que se utiliza la misma clave para cifrar y descifrar. Esto contrasta con el cifrado asimétrico, en el que hay claves separadas para el cifrado y el descifrado. Utiliza una clave de 256 bits, que proporciona un alto nivel de seguridad. Cuanto más larga sea la clave, más seguro será el cifrado, pero también puede requerir más recursos computacionales. En el modo CBC, cada bloque de texto plano realiza la operación XOR con el bloque de texto cifrado anterior antes del cifrado. Esto ayuda a garantizar que bloques de texto plano idénticos no den lugar a bloques de texto cifrado idénticos añadiendo una capa extra de seguridad.
El modo CBC requiere el uso de un vector de inicialización (IV), que es una entrada adicional al proceso de cifrado. El IV añade aleatoriedad al proceso de cifrado y ayuda a evitar patrones en el texto cifrado. Por esto lo consideramos seguro, además hay que tener en cuenta que se utiliza en varias aplicaciones para proteger datos confidenciales, incluido el cifrado de archivos y discos, protocolos de comunicación seguros como HTTPS.

Aunque algo menos usado, muy útil en criptoanálisis gracias a su diversificación de claves que es más compleja y tiene mayor cantidad de rondas, **Serpent** es un algoritmo de cifrado por bloques, mientras que **GCM** (Galois/Counter Mode) es un modo de funcionamiento para cifradores por bloques como Serpent.

Serpent funciona con bloques de datos de tamaño fijo (normalmente **128** bits) y utiliza una clave de distintas longitudes (por ejemplo, 128, 192 o 256 bits). Serpent es conocido por sus sólidas propiedades de seguridad y se considera muy seguro frente a diversos ataques criptográficos. El modo de funcionamiento GCM proporciona

tanto la funcionalidad de cifrado como la de código de autenticación de mensajes. GCM utiliza un vector de inicialización (IV) único para cada mensaje y lo combina con un contador para generar un flujo de datos pseudoaleatorios. A continuación, estos datos se realiza la operación XOR con el texto sin formato para su cifrado. Una de las características más notables de GCM es su capacidad para proporcionar integridad de datos además de cifrado. Se utiliza habitualmente en protocolos de comunicación segura como TLS (Transport Layer Security) y es favorecido por su eficacia y seguridad.

Por ser diseñado para tener un rendimiento rápido en software, **ChaCha20-Poly1305,** es una combinación de cifrado y autenticación ampliamente utilizada que proporciona seguridad y confidencialidad para la transmisión de datos. Suele emplearse en protocolos de comunicación segura como TLS (Transport Layer Security) y VPN (Virtual Private Networks).

ChaCha20 es un cifrado de flujo de clave simétrica diseñado por Daniel J. Bernstein. Es conocido por su simplicidad, alta velocidad y fuertes propiedades de seguridad. Funciona con una clave de 256 bits y un nonce (número utilizado una vez) de 64 bits para producir un flujo de datos pseudoaleatorios. ChaCha20 genera un flujo de claves que realiza una operación XOR con el texto plano para producir el texto cifrado, lo que lo hace adecuado para el cifrado de flujos.

Por otro lado Poly1305 es un algoritmo de autenticación que se utiliza para garantizar la integridad y autenticidad de los datos. Toma una clave de 256 bits y un mensaje como entrada y produce una etiqueta de autenticación de 128 bits (también conocida como MAC, o Código de Autenticación de Mensaje).

Cuando se combinan ChaCha20 y Poly1305, ChaCha20 se encarga del cifrado de los datos, mientras que Poly1305 proporciona autenticación y protección de la integridad.

Para probar estos algoritmos hemos cogido dos imágenes distintas y las hemos descifrado para comparar los tiempos entre ellos:

Cifrado	AES-256-CBC		Serpent-128-GCM		ChaCha20-Poly1305	
tam pequeño	0.755KB	1,46ms	11.6KB	15,3ms	11.6KB	6ms
tam grande	20,9KB	0.53ms	455KB	16,23ms	455KB	15,62ms
Descifrado	AES-256-CBC		Serpent-128-GCM		ChaCha20-Poly1305	
tam pequeño	0.755KB	0.47ms	11.6KB	27,7ms	11.6KB	15.66ms
tam grande	20.9kb	5,2ms	455KB	31,2ms	455KB	26,41ms

Adjunto el código usado para cifrar/descifrar en aes256 y las imágenes usadas.

Como salvaguardar la confidencialidad de la información

Usare carpetas o volúmenes de soporte físicos como discos duros, sobre todo me centrare en una herramienta que sea configurable.

Por lo dicho voy a usar Veracrypt, una herramienta configurable para analizar la salvaguarda de la confidencialidad de la información de carpetas o volúmenes.

El proceso de instalación es el siguiente:

- Usando un sistema Linux con Ubuntu en la versión 23.04:

Comienzo por descargar Veracrypt de su página oficial: https://www.veracrypt.fr/en/Downloads.html

Una vez descargado abro una consola en la carpeta donde se encuentra el archivo descargado, introducimos el siguiente comando para instalar el archivo descargado como super usuario sudo.

Sudo apt update

Sudo apt upgrade

sudo dpkg -i veracrypt-1.26.7-Ubuntu-23.04-amd64.deb

Si el paso 3 nos diese problemas introducimos el siguiente comando:

sudo apt -f install

Con esto completamos la instalación de Veracrypt en Ubuntu.

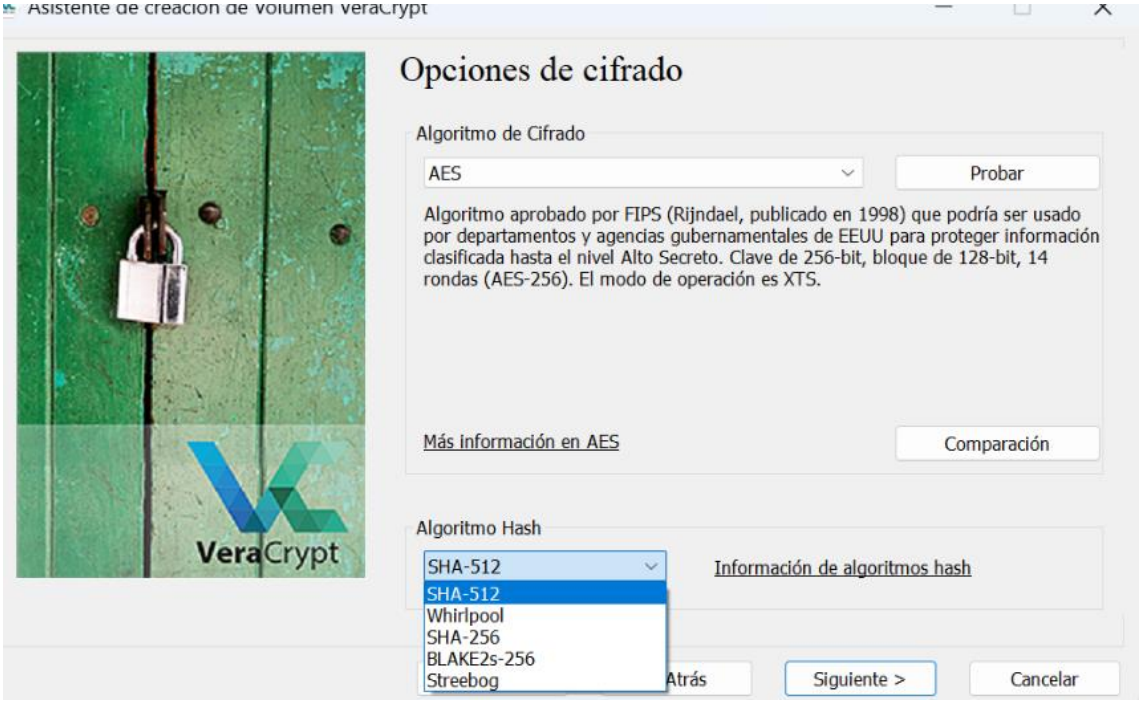
En mi caso tengo un doble boot con Windows 10 y Ubuntu, por lo que también lo instalare en Windows 10 , es el mismo proceso para Windows 11:

Descarga en: https://www.veracrypt.fr.

y ejecuto él .exe.

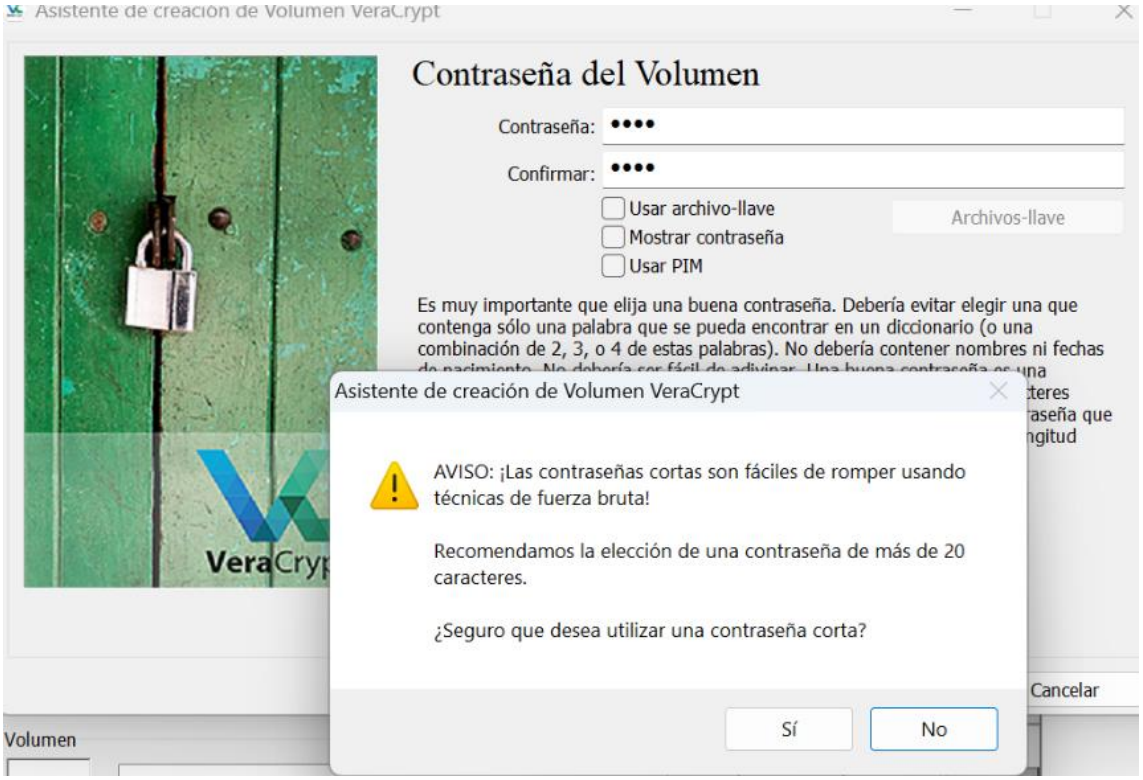
Respecto al uso de la aplicación, ejecutamos y clicamos en crear volumen, elegimos el lugar de nuestro PC donde guardar el volumen y pulsamos en siguiente con todas las opciones por defecto, he elegido el algoritmo de cifrado AES-256 con SHA-256. Le damos un tamaño de prueba de 100MB y por último establecemos una contraseñase nos avisa del peligro de una contraseña tan corta pero al ser una prueba no le damos importancia y seguimos adelante.

Adjunto imágenes del proceso:




Asistente de creación de Volumen VeraCrypt
Tipo de Volumen
Volumen VeraCrypt común
Seleccione esta opción si quiere crear un volumen VeraCrypt nor
Volumen VeraCrypt oculto
Podría suceder que alguien le obligara a revelar la contraseña de
cifrado. Hay muchas situaciones en las que no podría negarse a
contraseña (por ejemplo, debido a la extorsión). Usar el llamado
oculto" le permite resolver estas situaciones sin revelar la contra
volumen.
Más información acerca de volúmenes ocultos
VeraCrypt
Ayuda
< Atrás
Siguiente >


Asistente de Creación de Volúmenes VeraCrypt
Crear un contenedor de archivos cifrado
Crea un disco cifrado virtual dentro de un archivo. Recomendado para usuarios sin experiencia.
Más información
Cifrar partición/unidad secundaria
Cifra una partición en cualquier unidad interna o externa (ej: unidad flash). Opcionalmente, crea un volumen oculto.
Cifrar la partición/unidad del sistema entera
Cifra la partición/unidad donde Windows está instalado. Cualquiera que quiera acceder al sistema, leer y escribir archivos, etc. tendrá que introducir la contraseña antes de arrancar Windows. Opcionalmente, crea un sistema oculto.
Más información sobre cifrado del sistema
VeraCrypt
Ayuda
< Atrás
Siguiente >
Cancelar


Asistente de creación de Volumen VeraCrypt
Contraseña del Volumen
Contraseña:
Confirmar:
Usar archivo-llave
Mostrar contraseña
Usar PIM
Archivos-llave
Es muy importante que elija una buena contraseña. Debería evitar elegir una que contenga sólo una palabra que se pueda encontrar en un diccionario (o una combinación de 2, 3, o 4 de estas palabras). No debería contener nombres ni fechas
Asistente de creación de Volumen VeraCrypt
AVISO: ¡Las contraseñas cortas son fáciles de romper usando técnicas de fuerza bruta!
Recomendamos la elección de una contraseña de más de 20 caracteres.
¿Seguro que desea utilizar una contraseña corta?
Sí
No
Cancelar
Volumen


Asistente de creación de Volumen VeraCrypt
Formato del volumen
Opciones
Sistema de FAT
Cluster Por defec
Full Format
Dinámico
Pool Aleatorio:
Clave Cabecera:
Clave Maestra:
Abortar
Hecho
Velocidad
Quedan
IMPORTANTE: Mueva el ratón al azar todo lo posible dentro de esta ventana. Cuanto más lo mueva, mejor. Esto incrementa significativamente la fuerza criptográfica de las claves de cifrado. Luego haga clic en 'Formatear' para crear el volumen.
Aletoriedad Obtenida De Movimientos de Ratón
Ayuda
< Atrás
Formatear
Cancelar

Como podríamos recuperar la información almacenada?

En el caso de que tuviéramos por ejemplo un contenedor de información generado con la herramienta Veracrypt debemos tener en cuenta posibles casos, por ejemplo si hemos perdido todas las credenciales lo tenemos más complicado, más si le hemos puesto una contraseña sólida y fuerte, un cifrado fuerte como el AES y un hash SHA-512, pero pongámonos en un supuesto habitual, se ha perdido la contraseña pero recordamos parte de ella, usaría Hashcat. El comando:

hashcat -m 13721 -a 3 drive2.v -1?l?u abcd..H?10?d?d?d --self-test-disable

Como solución nos da la siguiente contraseña: abcd..HJ0000, probamos montar el volumen con esta contraseña y lo conseguimos.

Volviendo al primer caso habitual, decir que como he explicado antes el cifrado AES con una clave de 32 dígitos/letras/símbolos totalmente aleatorios proporciona un alto nivel de seguridad, sobre todo si lo aplicamos a una clave que sea lo suficientemente fuerte y no se haya comprometido, por lo que en ese caso no hay mucho más que hacer.

Primero deberíamos comprobar si las claves estaban almacenadas en el ordenador, en el caso de que no se encontrara ninguna información extra es poco probable que alguien pueda recuperar ese volumen.

Muy importante enfatizar en que las claves se deben gestionar de manera adecuada implementando unas políticas de seguridad sólidas, por ejemplo acceso restrictivo a las claves y su gestión centralizada, que las claves sean robustas y cambiadas cada cierto tiempo.

Salvaguardar información en forma de imágenes de forma sencilla

Para guardar en específico imágenes tenemos opciones más sencillas, en internet hay páginas que nos cumplen esta función, como:

https://encrypt.imageonline.co/index-es.php

https://decrypt.imageonline.co/

Con estas página puedes encriptar con la primera y desencriptar con la segunda las imágenes con contraseñas robustas con la longitud, y robustez que desee el usuario, ademas al ser online puede usarse desde el movil u ordenador desde cualquier sitio con acceso a internet, lo que es un detalle.

Ademas tiene la ventaja de ser de una empresa importante del sector con la seguridad de que es fiable.

Para la gestión de Claves:

Podemos usar las páginas mencionadas en la carpeta de “Prueba_de_obtencion_de_contraseñas”, en concreto https://www.lastpass.com/es/features/password-generator y para el almacenamiento seguro se puede emplear un "Key Management Service" (KMS) o una solución de almacenamiento seguro como HashiCorp Vault.

Adjunto una página de una empresa líder en el sector de la seguridad online que sirve justamente como gestor de contraseñas:

gestor_de_contraseñas

https://www.devglan.com/online-tools/text-encryption-decryption

La segunda ya es una página que hemos usado con anterioridad ya que puede servir para almacenar en local las contraseñas guardando la key y el encriptado en sitios distintos, ya que uno de los ataque más usuales viene por parte de brechas de seguridad físicas en nuestros empleados como apuntar las contraseñas en papeles etc, con esta página necesitan ambas para recuperar la contraseña:

El usuario elegiría una palabra como contraseña, la que el prefiera, por ejemplo patata, con la página la encriptaría y usaría una palabra clave la cual puede ser por ejemplo frita, si en algún momento se le olvida la palabra que el uso para encriptar, que sería patata, con la clave en este caso frita y el encriptado que debe tener guardado en un lugar seguro puede recuperarla donde quiera, osea le serviría como seguridad por si le roban la palabra guardada encriptada ya que no podrían hacer nada con ella sin la clave que es frita.

Ambas páginas se pueden combinar para dar un paso más en seguridad.

Consideraciones adicionales:

La gestión de acceso es otro punto crítico**:**

Implementa un sistema de control de acceso robusto que permita asignar permisos específicos a médicos autorizados. Solo aquellos con permisos adecuados deben tener acceso a las claves de cifrado.

Rotación de Claves:

Es muy recomendable establecer una política de rotación de claves. Las claves deben ser rotadas periódicamente para limitar la exposición en caso de una violación de seguridad.

Capacitación del Personal**:** Advertir al cliente y concienciarle para proporcionar capacitación al personal médico sobre las mejores prácticas de seguridad y la importancia de proteger las claves de cifrado, ya que como he dicho antes una gran parte de los ataques vienen por el entorno físico y brechas de seguridad humanas.

Este enfoque proporciona una capa de seguridad sólida para asegurar la confidencialidad de las imágenes médicas y garantiza que solo el personal médico autorizado tenga acceso a la información sensible.

Almacenamiento seguro de claves con Python

Para almacenar de manera segura las claves usando Python he desarrollado un script que implementa un almacén seguro de claves (no utiliza uno de los que ya implementa el SO) y mediante la clave que se encuentra en dicho almacén, generamos un objeto Fernet para encriptar la imagen que nos piden.

Fernet es una librería de Python que utiliza el algoritmo AES-256 en modo CBC y una función MAC para verificar la integridad de los datos.

La clave maestra se almacena en un archivo llamado clave_maestra.key que es lo suficientemente seguro para no poder abrirlo como tal ya que se va autogenerando por cada ejecución al utilizar un salt aleatorio.

Un problema de seguridad del código es que la contraseña utiliza se encuentra en el mismo, pero eso es para que sea ilustrativo, eso es tan fácil como almacenarlo en un .txt y luego leerlo de ahí.

Adjunto incluyo todos lo necesario para realizar las pruebas, lo único que hay que hacer es ejecutar el archivo Python que se encuentra en la carpeta y se generaran dos archivos jpg, uno con la imagen encriptada y otra con la desencriptada. También aparece el archivo clave_maestra.key, que no se puede abrir.

Almacenamiento seguro de claves con Java

Las diferentes opciones para el almacenamiento de claves en Java son:

KeyStore de Java: Java proporciona KeyStore como un almacén de claves por defecto. Para almacenar claves y certificados se puede utilizar JKS (Java KeyStore) o PKCS12 (Portable KeyStore). Son adecuados para aplicaciones Java estándar. El proceso de uso implica cargar y almacenar claves y certificados en un archivo KeyStore, que está protegido por una contraseña.

HSM (Hardware Security Module): Los módulos de seguridad hardware son dispositivos físicos diseñados para la gestión segura de claves. Java ofrece soporte para integrar HSM en aplicaciones mediante proveedores de seguridad JCE (Java Cryptography Extension) que admiten HSM. Esto es adecuado para aplicaciones críticas de seguridad.

Bóvedas de Claves en la Nube (AWS KMS, Azure Key Vault, Google Cloud KMS): Estos servicios en la nube permiten el almacenamiento y gestión de claves en un entorno altamente seguro. Se puede acceder a ellos desde aplicaciones Java a través de las SDK correspondientes. Son ideales para aplicaciones en la nube y servicios web.

Librerías de Almacenamiento de Claves (Keychain, Vault, etc.): librerías de almacenamiento de claves de terceros que proporcionan características avanzadas de gestión de claves y seguridad.

Vista las opciones realizo la elección en base a los requisitos, para el sistema propuesto en esta práctica determino que el más adecuado será el KeyStore de Java.

En el archivo adjunto queda reflejado tanto el código utilizado como los resultados de las pruebas (imagen cifrada y descifrada). El código se divide en dos partes principales: configuración del almacén de claves y cifrado-descifrado de la imagen.

Primero, creo el keyStore con la herramienta keytool de Java e introduzco el siguiente código por consola:

keytool -gensecKey -alias **nuestro_alias** -keyalg AES -keysize 256 -keystore keystore.jks

Una vez creado, el código funcionará sin problemas, primero configura el almacén de claves, después usando FileInputStream se carga el almacén de claves, la contraseña se proporciona para desbloquear el almacén y acceder a la clave privada almacenada en él.

La clave privada se recupera del almacén de claves utilizando el alias **clave1** y la contraseña asociada, después genera un vector de inicialización de números aleatorios, esencial para el cifrado y descifrados con el modo CBC para AES.

En la segunda fase creamos la instancia de Cipher para realizar las operaciones de cifrado. La inicialización se realiza en modo de cifrado usando la clave privada y el vector de inicialización. Se combinan los datos del cifrados con los del vector de inicialización.

Por último se descifra y se guarda la imagen, para el descifrado se extrae el vector de inicialización de los primeros 16 bytes de los datos cifrados. Se inicializa la instancia Cipher en modo descifrado usando la misma clave privada y el vector de inicialización.