GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

Unidad 4.2
Criptografía

Índice

Licencia

1 Cifrado asimétrico

El cifrado asimétrico, un pilar fundamental en la protección de la información en el ámbito digital, ha transformado la manera en que garantizamos la seguridad y confiabilidad de las comunicaciones electrónicas. Su innovador enfoque se basa en el uso de un par de claves únicas pero interconectadas, mejorando así el mecanismo de cifrado simétrico. Más allá de su contribución a la privacidad, el cifrado asimétrico desempeña un papel esencial en la autenticación y la firma digital. Este sistema proporciona una capa de seguridad crítica en transacciones en línea, verificación de identidades digitales y la integridad de documentos electrónicos, destacando su versatilidad y relevancia en el panorama de la seguridad cibernética.

1.1 Introducción a los conceptos clave

A diferencia del cifrado simétrico, donde una única clave se utiliza tanto para cifrar como para descifrar, en el cifrado asimétrico se introduce un nivel adicional de complejidad y seguridad al utilizar dos claves:

- **Llave pública** (todo el mundo la puede ver)
- **Llave privada** (no se debe compartir con nadie)

Con este modelo el destinatario necesita dos llaves: la clave pública y la calve privada para descifrar un mensaje. Esta dualidad en las claves fortalece significativamente la seguridad del proceso, ya que incluso si la clave pública está disponible públicamente, el descifrado sigue siendo imposible sin la clave privada correspondiente.

1.2 El par de llaves del cifrado asimétricos

En el cifrado asimétrico, el par de llaves, compuesto por la pública y privada, configura un sistema de cifrado accesible y seguro. Veremos cómo estas llaves, aunque vinculadas, desempeñan funciones diferenciadas en el proceso de cifrado y descifrado, elevando así el nivel de seguridad en las comunicaciones digitales.

Cómo se genera el par de claves:

El par de claves en el cifrado asimétrico se genera mediante **algoritmos matemáticos especializados**. Un ejemplo común es el algoritmo RSA (Rivest-Shamir-Adleman). En este proceso, se elige un par de números primos aleatorios bastante grandes. Estos números se utilizan para calcular otros valores, y a partir de estos cálculos **se obtiene el par de claves** (pública y privada).

La seguridad del proceso:

La seguridad del cifrado asimétrico radica en la **dificultad de realizar la operación inversa**, es decir, deducir la clave privada a partir de la clave pública. Aunque la generación del par de claves es un proceso relativamente sencillo y rápido, la operación de factorización inversa necesaria para obtener la clave privada a partir de la clave pública resulta enormemente compleja. **Este desequilibrio en la complejidad de las operaciones garantiza** que, mientras sea factible para el remitente cifrar el mensaje con la clave pública, descifrarlo sin la clave privada correspondiente resulta computacionalmente prohibitivo y prácticamente imposible.

Tamaño de las claves:

El tamaño de las claves en el cifrado asimétrico es un factor crucial para la seguridad. A medida que aumenta el tamaño de las claves, se incrementa la complejidad computacional necesaria para realizar ataques de fuerza bruta o de factorización. Las claves privadas y públicas son mucho más grandes que cualquier clave de cifrado simétrico:

- **Clave simétrica**: tamaño mínimo recomendado de 256.
- **Clave pública y privada**: tamaño mínimo recomendado de 1024 para cada clave

Como podéis ver en la actualidad, el tamaño mínimo recomendad para las claves asimétricas es de 1024 bits, muchas implementaciones avanzadas utilizan longitudes de clave de 2048 bits o incluso más.

1.3 Uso de la Clave Pública en Criptografía

En el contexto de la criptografía asimétrica, la "clave pública" se presenta como un componente central. Esta clave, conocida y compartida abiertamente, es la piedra angular para entender tanto el cifrado de mensajes como la firma digital.

- **Clave pública en el cifrado de mensajes:**

  El propósito del cifrado de clave pública es salvaguardar la confidencialidad del mensaje transmitido. Para comprender su funcionamiento, podemos emplear la analogía del buzón de casa. La ranura está expuesta a mucha gente y su ubicación (la dirección de la calle) es conocida por muchas personas (**clave pública**). El acceso al contenido del buzón, simbolizado por la capacidad de abrirlo, está restringido únicamente al propietario, **quien posee la llave privada**. Este proceso garantiza que solo el destinatario (a quien va dirigida la carta) pueda descifrar y acceder al mensaje confidencial.

- **Clave pública en la firma digital:**

La calve pública se utiliza para garantizar la autenticidad del mensaje. Si lo comparamos con los sellos de lacre de la antigüedad, el sello del rey era ampliamente conocido y ya antes de abrir la carta, se sabía quien era el remitente del mensaje. La firma digital, al igual que el sello de lacre, no sólo te garantizaba el remitente, sino que también garantizaba que el mensaje no se ha manipulado en el camino.

(*Nota histórica*: Antiguamente las cartas se sellaban con lacre, que se aplicaba en caliente a la carta y se sellaba en ese momento. Una vez enfriado, el lacre se volvía completamente rígido, y su ruptura era irreversible. Esta práctica garantizaba que la carta no había sido manipulada previamente, preservando la integridad de su contenido y el sello permitía identificar y garantizar el remitente.)

1.4 Cifrar un mensaje

A continuación se tratarán todos los pasas de forma detallada que hacen falta para cifrar y descifrar un mensaje usando criptografía asimétrica:

- Ana quiere enviar un mensaje cifrado a David. Se trata de información confidencial y no quiere que nadie más lo lea.
- Ana **conoce la clave pública de David** (la conoce mucha gente). David creó sus claves (pública y privada) con el algoritmo RSA. David envió su clave pública a varias personas.
- Ana cifra el mensaje usando un algoritmo de tipo PKCS-OAEP (algoritmo que funciona muy bien con claves generadas con RSA).
- Ana envía el mensaje a David.
- David recibe el mensaje encriptado.
- David usa el algoritmo PKCS-OAEP y su cl**ave privada para desencriptar** el mensaje.
- David puede leer el mensaje que le ha enviado Ana.

1.5 Firmar un mensaje

En este punto abordaremos de manera detallada el proceso técnico de firmar un mensaje:

- David quier enviar su Currículo a Ana:
  - A David no le importa que otra persona lea este mensaje. Si le gusta su CV a lo mejor le contrata.
  - David quiere garantizar a Ana que ese Currículo es suyo.
  - David quiere garantizar que nadie modifica su Currículo con malas intenciones (para que no le contraten en la empresa de Ana).
- David hace un Hash del documento con SHA256. Así si alguien lo modifica el Hash será diferente.
- David **usa su clave privada para encriptar el Hash** de su Curriculo usando el algoritmo PKCS-PSS.
- David envía a Ana su Currículo y el Hash encriptado (esto es la firma)
- Ana recibe el Currículo y el Hash encriptado.
- Ana vuelve a hacer el Hash del Currículo de David. Ahora usa el algoritmo PKCS-PSS para confirmar que el Hash es igual al Hash firmado por David. Para que esto funcione ha de pasar al algoritmo también **la clave pública de David**.
- El algoritmo indica a Ana que todo es correcto. Ahora Ana sabe que **el currículo es de David** (ha usado su clave pública) y que **nadie lo ha modificado** (la comparación de los 2 Hash ha sido correcta).

1.6 Desafíos en la Gestión de Claves Públicas

La confianza en la autenticidad de las claves públicas en entornos digitales plantea desafíos cruciales. Una de las cuestiones fundamentales es cómo verificar que la clave pública de un usuario realmente pertenece a ese usuario. Además, surge la interrogante sobre cómo hacer que esta clave esté disponible para otros usuarios o servicios. Dos enfoques comunes para abordar estos problemas son mediante Infraestructuras de Clave Pública (PKI) y la construcción de una red de confianza.

Infraestructuras de Clave Pública (PKI):

Las PKI son estructuras organizativas que emiten certificados de usuario, publicando sus claves públicas y asegurando su autenticidad (junto con atributos adicionales). Las entidades encargadas de emitir estos certificados se denominan Autoridades de Certificación (CA). Este enfoque brinda una base formal para establecer y validar la confianza en las claves públicas, respaldando la seguridad en las transacciones digitales.

Red de Confianza:

En este enfoque, los usuarios optan por establecer una red de confianza personal. Por ejemplo, si no conocen directamente a un usuario, pueden confiar indirectamente en su clave pública a través de conexiones de confianza previas. Un modelo notable que emplea esta estrategia es PGP (Pretty Good Privacy). En este contexto, la confianza se construye a través de relaciones interpersonales, donde la validación de las claves públicas se basa en la garantía proporcionada por conocidos mutuos, fortaleciendo así la seguridad en la comunicación digital.

1.7 Ventajas y desventajas del cifrado asimétricos

El cifrado asimétrico ofrece beneficios clave en términos de seguridad y funcionalidad, pero también presenta desafíos en cuanto a eficiencia computacional y tamaño de claves. La elección entre cifrado asimétrico y simétrico a menudo depende de las necesidades específicas de seguridad y los requisitos de rendimiento de una aplicación o sistema.https://es.wikipedia.org/wiki/Firma_digital

Ventajas del Cifrado Asimétrico:

- **Seguridad en la Comunicación Pública:**
  - **Ventaja**: La clave pública puede ser compartida abiertamente, permitiendo a cualquier entidad cifrar mensajes para el propietario de la clave privada.
  - **Implicación**: Facilita la seguridad en comunicaciones públicas, ya que la clave de cifrado puede ser distribuida sin comprometer la clave de descifrado.
- **Autenticación y Firma Digital**:
  - **Ventaja**: Permite la autenticación de la fuente y la firma digital de documentos.
  - **Implicación**: Facilita la verificación de la autenticidad de mensajes y documentos, ofreciendo una capa adicional de seguridad en transacciones digitales.
- **Gestión de Claves más Sencilla**:
  - **Ventaja**: No es necesario compartir una clave secreta entre partes comunicantes.
  - **Implicación**: Simplifica la gestión de claves, ya que cada entidad solo necesita mantener segura su clave privada.

Desventajas del Cifrado Asimétrico:

- **Tamaño de las Claves:**
  - **Desventaja**: Las claves asimétricas suelen ser más largas que las simétricas.
  - **Implicación**: Mayor uso de recursos para almacenar y transmitir claves, así como para realizar operaciones criptográficas.
- **Menor Eficiencia en Cifrado de Grandes Cantidades de Datos**:
  - **Desventaja**: No es tan eficiente como el cifrado simétrico cuando se trata de grandes volúmenes de datos. Los algoritmos asimétricos son computacionalmente más costosos que sus contrapartes simétricas.
  - **Implicación**: Puede no ser la mejor opción para cifrar grandes archivos o transmisiones continuas de datos.

2 La criptografía híbrida

La criptografía híbrida es un enfoque que combina las fortalezas del cifrado asimétrico y simétrico para abordar eficientemente los desafíos de la seguridad en la transmisión de información.

2.1 Proceso de la Criptografía Híbrida

1. **Generación de Claves Asimétricas:**

   Inicialmente, se genera un par de claves asimétricas para cada usuario: una clave pública, que se comparte abiertamente, y una clave privada, que se mantiene en secreto.

2. **Cifrado Simétrico de Datos:**

   Para la transmisión de datos, se utiliza un algoritmo de cifrado simétrico más eficiente en términos computacionales. Se genera una clave de sesión única y aleatoria para cada transacción o sesión de comunicación.

3. **Cifrado de la Clave Simétrica con Clave Pública:**

   La clave simétrica generada se cifra utilizando la clave pública del destinatario. Esta operación es rápida y eficiente, ya que implica cifrar una cantidad mínima de datos (la clave simétrica) con la clave pública.

4. **Envío de Datos Cifrados Simétricamente y Clave Cifrada Asimétricamente:**

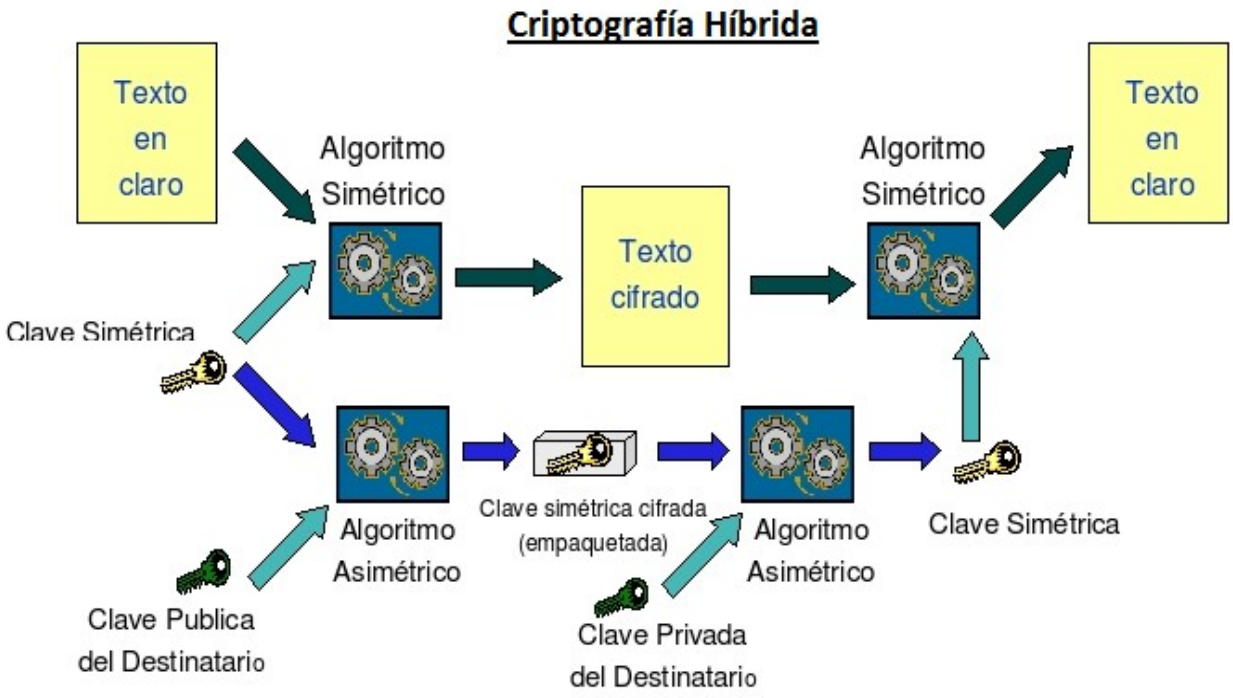
   Los datos, cifrados simétricamente, y la clave de sesión cifrada asimétricamente se envían al destinatario.

5. **Descifrado de la Clave Simétrica con Clave Privada:**

   El destinatario utiliza su clave privada para descifrar la clave simétrica. Como la clave privada solo la conoce el destinatario, se garantiza la confidencialidad de la clave simétrica.

6. **Descifrado Simétrico de Datos:**

   La clave simétrica descifrada se utiliza para descifrar los datos transmitidos simétricamente, restaurando así el mensaje original.



2.2 Ventajas de la Criptografía Híbrida

- **Eficiencia Computacional:**

  Aprovecha la eficiencia del cifrado simétrico para el manejo de grandes volúmenes de datos.

- **Seguridad Asimétrica:**

  Aprovecha la seguridad del cifrado asimétrico para la transmisión segura de claves simétricas, superando las limitaciones de eficiencia de este último.

- **Adaptabilidad:**

  Permite ajustar el tamaño de la clave simétrica según las necesidades de seguridad, sin afectar la eficiencia del cifrado asimétrico.

3 Algoritmo RSA para encriptar y desencriptar un fichero.

El algoritmo RSA, acrónimo de sus creadores Rivest, Shamir y Adleman, ha sido un pionero en la criptografía de clave pública desde su desarrollo en 1979 y su posterior patentación por el MIT en 1983. La robustez de este algoritmo se basa en el desafiante problema de la factorización de números enteros. RSA es reconocido como el primer sistema de seguridad digital de clave pública y ha sido una pieza esencial en la evolución de los algoritmos de cifrado. A lo largo del tiempo, ha coexistido con alternativas como el Algoritmo de Firma Digital (DSA), creado por la Agencia de Seguridad Nacional (NSA) en 1991. Ambos, RSA y DSA, son algoritmos versátiles que han acabado siendo cruciales en el despliegue de algoritmos de cifrado. Ambos pueden emplearse en el entorno cliente-servidor.

Se sostiene que la seguridad de RSA perdurará mientras no existan métodos eficaces para factorizar rápidamente números grandes en productos de primos. Aunque se especula que la computación cuántica podría presentar soluciones a este problema de factorización, algunos investigadores expresan dudas sobre la obsolescencia inminente de estos algoritmos.

Características de RSA

RSA destaca por su versatilidad y eficiencia en el cifrado y la firma de documentos. Aunque más rápido en estas operaciones, el proceso de generación del par de claves es más lento. Se recomienda un tamaño mínimo de clave de 2048 bits para garantizar una seguridad óptima. Este algoritmo se emplea tanto para cifrar documentos completos como para la firma de archivos.

3.1 Otros algoritmos de encriptación asimétrica

Características de DSA:

El Algoritmo de Firma Digital (DSA) se destaca por su rapidez en las operaciones de desencriptación y verificación. Se recomienda un tamaño mínimo de clave de 512 bits. Su uso principal radica en la firma de documentos.

Elliptic Curve Cryptography (ECC):

ECC, otro algoritmo de encriptación asimétrica, ofrece una alternativa eficiente a RSA y DSA. Aprovecha las propiedades matemáticas de las curvas elípticas para proporcionar la misma seguridad con claves más cortas. Este enfoque se vuelve especialmente relevante en aplicaciones con restricciones de recursos, como dispositivos móviles o sistemas con potencia de procesamiento limitada.

Elección y Consideraciones:

La elección entre RSA, DSA y ECC dependerá de las necesidades específicas del sistema y las restricciones de recursos. Mientras RSA y DSA se mantienen como opciones sólidas, ECC destaca por su eficiencia en términos de espacio y rendimiento.

3.2 Utilización de RSA en Python

Cuando seleccionamos un algoritmo, es crucial considerar que estos suelen trabajar en conjunto con otros para realizar sus funciones específicas de encriptación o firma. Al optar por RSA, es necesario tener presente las recomendaciones proporcionadas por la API de la librería de Python que estamos utilizando.

En nuestro caso, utilizamos la librería pycryptodom, por lo que es esencial revisar las indicaciones detalladas en el sitio web oficial: https://www.pycryptodome.org/src/api.

Dicho módulo ofrece dos opciones para la generación del par de claves: RSA o ECC.

En esta unidad usaremos el algoritmo RSA y por ello habrá que generar primero un par de claves. Código que se propone para hacerlo:

```
from Crypto.PublicKey import RSA
from pathlib import Path

key = RSA.generate(2048) #creamos la llave privada RSA

private_key = key.exportKey() #codifica la clave en un formato

fichero_path = Path(__file__).parent / "privada_servidor.pem" #

file_out = open(fichero_path,"wb")
file_out.write(private_key) #escribimos la clave formateada en
file_out.close()

public_key = key.publickey().exportKey() #generamos la clave pú

fichero_path = Path(__file__).parent / "publica_servidor.pem" #

file_out = open(fichero_path,"wb")
file_out.write(public_key) #escribimos la clave pública en otro
file_out.close()
```

Para practicar con el algoritmo RSA se va a plantear el ejercicio de enviar un documento encriptado destinado a una persona llamada “A”. Para encriptar el documento se usará criptografía híbrida. Recordamos los pasos básicos para realizar una encartación híbrida.

1. **Generación de Claves Asimétricas:**
   Crear un par de claves asimétricas (pública y privada) mediante un algoritmo de criptografía asimétrica como por ejemplo RSA. **(Hecho)**
2. **Generación de Clave Simétrica Aleatoria:**
   Crear una clave simétrica única y aleatoria para la encriptación simétrica.
3. **Encriptación del Documento con Clave Simétrica:**
   Utilizar la clave simétrica generada para encriptar todo el contenido del documento.
4. **Encriptación de la Clave Simétrica con Clave Pública RSA:**
   Encriptar la clave simétrica con la clave pública RSA de del destinatario mediante el algoritmo asimétrico RSA.
5. **Envío del Documento Encriptado y Clave Simétrica Encriptada:**
   Enviar tanto el documento encriptado como la clave simétrica encriptada al destinatario.

Concretamos estos pasos (del **2** al **4**) para Python:

- Los algoritmos tratan los datos en binario y por ello habrá que usar las ya conocidas funciones: encode() o encode(“utf-8”).
- Para generar una **Clave Simétrica Aleatoria** usaremos la función:
  get_random_bytes()
- Se usará el **algoritmo AES para la encriptación simétrica** del documento. El módulo de Pycryptodome así lo recomienda. El algoritmo AES nos devuelve el texto cifrado y el **Hash** (aquí al Hash se le llama **Tag**)
- Ahora se necesita la clave pública RSA de la persona A para encriptar la **Clave Simétrica** (En el código se la llama **clave de sesión**).
- Para cifrar la Clave Simétrica (clave de sesión) se usará un algoritmo de cifrado asimétrico que use claves RSA. La librería Pycryptodom recomienda usar en este caso el algoritmo de cifrado asimétrico PKCS1_OAEP, que combina bien con claves RSA.

Paso 5: Envío del Documento Encriptado y Clave Simétrica Encriptada

Una vez tenemos toda la información encriptada se abrirá un fichero para guardar el resultado de encriptar el documento. Se escribirá en este orden y luego hay que asegurarse que la persona “A” lea el fichero en este mismo orden o no podrá descifrar el mensaje:

1. La clave de sesión encriptada con la **clave pública RSA de la persona A** (para hacer esto hemos usado el algoritmo de cifrado asimétrico PKCS1_OAEP)
2. Otra clave aleatoria de seguridad llamada nonce.

   Es un número que genera AES para garantizar que las comunicaciones antiguas no se puedan reutilizar en ataques de playback.
3. El Hash del texto llamado tag
4. El texto encriptado con la calve de sesión (cifrado simétrico)

Código Python:

```
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
from pathlib import Path

data = ('Comentar el código es como limpiar el cuarto de baño;
        'pero el resultado es siempre una experiencia más agrad

datosBin = data.encode("utf-8") #pasamos el texto a binario par

session_key = get_random_bytes(16) #clave generada de forma ale

#Cifrado Simétrico: Encriptar los datos con AES
cipher_aes= AES.new(session_key, AES.MODE_EAX)
cipher_text, tag = cipher_aes.encrypt_and_digest(datosBin) #Dev

#Cifrado asimétrico
fichero_path = Path(__file__).parent / "publica_usuario_A.pem"
recipient_key = RSA.import_key(open(fichero_path).read()) #leem

#Encriptar la Clave Asimétrica con la clave pública el usuario_
cipher_rsa = PKCS1_OAEP.new(recipient_key) #creamos un nuevo ob
enc_session_key = cipher_rsa.encrypt(session_key)

#Guardamos toda la información en un fichero de salida, que es

fichero_path = Path(__file__).parent / "DatosEncriptados.bin" #
file_out = open(fichero_path,"wb")

file_out.write(enc_session_key)
file_out.write(cipher_aes.nonce) #es de 16 bytes y nonce es un
file_out.write(tag) #es de 16 bytes
file_out.write(cipher_text)
file_out.close()
```

Cuando la persona A recibe el fichero, este contiene: la clave de sesión encriptada (asimétrico), una clave nonce, el hash y el texto encriptado (simétrico).

Pasos que hay que hacer para desencriptar:

1. Abrir la **llave privada RSA** (es la de la persona A).
2. Leer y guardar en variables el contenido del documento.
3. Crear un objeto con el mismo algoritmo de cifrado, en este caso PKCS1_OAEP.
4. Desencriptar con el algoritmo PKCS1_OAEP la clave de sesión, usando la clave privada RSA
5. Con la calve de sesión ya desenciptada, podemos usar el algoritmo AES con el fin de desencriptar el texto (cifrado simétrico).
6. Al crear el objeto AES, le pasamos la clave de sesión y el número nonce (evita la reutilización)
7. En una misma operación desencriptamos el texto y comprobamos con el Hash que no ha sido modificado.
8. El texto esta en binario, por ello hay que decodificarlo: decode() o decode("utf-8")

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES, PKCS1_OAEP
from pathlib import Path

fichero_path = Path(__file__).parent / "DatosEncriptados.bin"

file_open = open(fichero_path,"rb") #abrimos el fichero encript

fichero_path = Path(__file__).parent / "privada_usuario_A.pem"

private_key = RSA.import_key(open(fichero_path).read()) #leemos

enc_session_key = file_open.read(private_key.size_in_bytes())
nonce = file_open.read(16) #leemos el número generado aleatoria
tag = file_open.read(16) #leemos el Hash del texto cifrado
cipherTexto = file_open.read() #Ya lo que queda en el fichero e

# Desencriptar la sesión RSA con la clave privada del usuario
cipher_rsa = PKCS1_OAEP.new(private_key) #Creamos un nuevo obje
session_key = cipher_rsa.decrypt(enc_session_key)

#Desenecriptamos los datos que se habían encriptado con el algo
cipher_aes = AES.new(session_key, AES.MODE_EAX, nonce)
data = cipher_aes.decrypt_and_verify(cipherTexto,tag) #Compara
print(data.decode("utf-8"))
```

4 Firma digital

En el contexto de la firma digital, se recurre comúnmente al algoritmo DSA. Sin embargo, en la librería pycryptodom, esta funcionalidad también se implementa utilizando el algoritmo RSA, aunque con una velocidad de ejecución ligeramente inferior respecto al DSA.

Pasos a seguir para firmar un documento:

1. Abro el fichero a firmar y genero el Hash de dicho fichero:

   La librería pycryptodom recomienda usar el **algoritmo SHA256** para generar el Hash

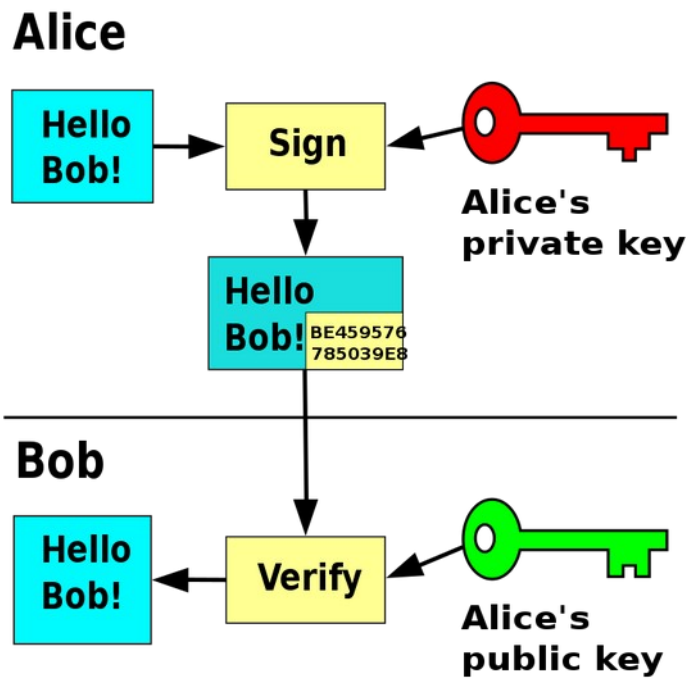
2. Encripto el Hash del documento con mi **clave privada RSA** :

   La librería pycryptodom recomienda usar el algoritmo PKCS1_PSS para generar firmas combinadas con claves RSA.

3. Guardo mi el Hash encriptado (la signatura) en un fichero: “SignaturaFichero.txt”.

4. Envío el documento **sin cifrar** y mi signatura por correo a una empresa. (Me pedían cierta documentación firmada)

Nota: Podría también haber encriptado el documento con la clave pública de la empresa y enviar así el documento encriptado y signado por mi.



4.1 Verificar un documento firmado

Cuando recibes un documento firmado, debes verificar el documento con su firma. En este caso tenemos un documento firmado por la persona A.

Pasos a seguir para comprobar el firma:

1. Abro la clave pública RSA de la persona A.
2. Abro el fichero que me ha enviado la persona A y creo el Hash del fichero usando el algoritmo SHA256. Uso SHA256 porque es el mismo algoritmo Hash que ha usado la persona A y que recomienda la librería pycryptodom.(Le diré Hash comprobación)
3. Abro el fichero que contiene la firma, en este caso: **SignaturaFichero.txt**.
4. Uso el mismo algoritmo de encriptación-desencriptación que la persona A.
    - En este caso la librería pycryptodom había recomendado usar el algoritmo PKCS1_PSS
    - Al algoritmo PKCS1_PSS le paso los siguientes parámetros:
        - **La clave pública RSA** de la persona A
        - **El código Hash** que he calculado yo (Hash comprobación)
        - **La firma** que me ha enviado la persona A (es el Hash del mismo fichero pero encriptado con la clave privada de la persona A)
5. El algoritmo PKCS1_PSS me dirá:
    - Si todos los datos son correctos (autentifica el documento y la persona)
    - Generará una excepción si algo falla.

Si todo es correcto podemos usar el documento, teniendo la tranquilidad que es de la persona A y que nadie lo ha manipulado.

5 Referencias

https://es.wikipedia.org/wiki/Criptograf%C3%ADa_asim%C3%A9trica

https://es.wikipedia.org/wiki/Infraestructura_de_clave_p%C3%BAblica

https://es.wikipedia.org/wiki/Autoridad_de_certificaci%C3%B3n

https://es.wikipedia.org/wiki/Certificado_de_clave_p%C3%BAblica

https://es.wikipedia.org/wiki/Red_de_confianza

https://es.wikipedia.org/wiki/Pretty_Good_Privacy

https://es.wikipedia.org/wiki/RSA

https://es.wikipedia.org/wiki/Firma_digital