Titulación: GRAD GENIERIA INFORMATICA Año Académico: 2024/25 Asignatura: Criptografía y seguridad informática Memoria de Prácticas 2	uc3m | Universidad Carlos III de Madrid

Grupo: 80

ID de grupo de prácticas: 2

Nombre de todos los alumnos: Jorge Agramunt Muro y Héctor Álvarez Marcos

Correo de todos los alumnos: 100495764@alumnos.uc3m.es y 100495794@alumnos.uc3m.es

Repositorio de código (enlace) si lo hubiera: https://github.com/CodificandoAlonso/P1_CRIPTO

Propósito de la Aplicación

La aplicación es un sistema de mensajería y gestión de productos que permite a los usuarios registrarse, iniciar sesión, agregar productos para la venta, comprar productos y enviar mensajes cifrados entre usuarios.

Uso de la firma digital

Utilizamos la firma digital para garantizar autenticidad: encriptando con la clave privada del usuario que envía el mensaje; integridad: garantizando que el contenido del mensaje no ha sido alterado después de haber sido firmado; y no repudio: impidiendo que el usuario que firma el mensaje pueda negar haber firmado el mensaje.

Algoritmos utilizados y por qué

Estos son los algoritmos que hemos utilizado para el intercambio y firma de mensajes

- **RSA**:
  - Usado para generar claves públicas y privadas, cifrado y firma digital.
  - Claves de 4096 bits se utilizan para mayor resistencia contra ataques futuros.
- **SHA-256 (Secure Hash Algorithm 256 bits)**:
  - Usado para calcular el hash de los datos antes de firmarlos digitalmente.
  - Es seguro y resistente a colisiones.
- **HMAC (Hash-based Message Authentication Code)**:
  - Usado para validar la integridad de claves y mensajes en combinación con claves simétricas.

Gestión y almacenamiento las claves y las firmas

- **Gestión de claves:**
  Las claves privadas generadas para cada usuario, servidor y autoridades son claves RSA. Se almacenan en archivos .pem en directorios específicos (keys/username/).

  Las claves públicas son compartidas como parte de los certificados o directamente como archivos .pem.

  Las claves simétricas están generadas por Fernet, cosa que se explica en la primera parte de la práctica.

  Gestión de firmas:
  A la hora de intercambiar mensajes, utilizamos este esquema de intercambio:
  $K_u\left(K_s\left(M,\ K_p\left(h(M)\right)\right)\right)$
  Ku = Clave pública
  Ks = Clave simétrica
  Kp = Clave privada
  h(M) = Hash del mensaje
  M = Mensaje

  Las firmas digitales se generan usando las claves privadas y se verifican con las claves públicas correspondientes.

  Almacenamiento:
  - Claves privadas: Archivos .pem cifrados almacenados en rutas seguras (keys/).
  - Claves simétricas: Almacenadas encriptadas dentro de archivos JSON protegidos mediante ChaCha20-Poly1305 .
  - Firmas: Incluidas como parte de los mensajes que siguen este esquema:
    $K_u\left(K_s\left(M,\ K_p\left(h(M)\right),\ token\right)\right)$
    Ku = Clave pública
    Ks = Clave simétrica
    Kp = Clave privada
    h(M) = Hash del mensaje
    M = Mensaje
    Token = número necesario para hacer el HMAC
    A la hora de guardar estos mensajes encriptados, nos hemos encontrado con el problema de que la clave pública puede encriptar un mensaje de hasta 446 bytes,

pero sólo la firma del hash tiene un tamaño de 512 bytes. La solución que encontramos a este problema fue partir la firma del hash en dos partes. Encriptamos por un lado el mensaje con el token del hash que el receptor pueda hacer el hash del mensaje y por el otro lado encriptar la firma partida en dos, lo que nos deja tres partes separadas para encriptar y luego guardarlo concatenado. A la hora de cargar los mensajes, cogemos la concatenación, la volvemos a separar, desencriptamos usando la clave privada y volvemos a concatenar, así tenemos sólo la encriptación simétrica. Ahora quedaría desencriptar el cifrado simétrico, quitar la firma del hash del mensaje, hacer el hash del mensaje y hacer la comparación.

Generación de los certificados de clave pública

Los certificados de clave pública se generan siguiendo estos pasos:

a. **Generación de claves RSA**: Se utiliza un par de claves RSA (privada y pública) para cada entidad (FASA, MVSA, y CSSA).
b. **Creación del certificado**: Usando la clave pública, se define los atributos (País, Cuidad, Nombre de la entidad...), y se firman utilizando la clave privada de la autoridad certificadora superior. En el caso de la entidad raíz, se firma a sí mismo.
c. **Almacenamiento**: Los certificados generados se guardan como archivos .pem.

Ejemplo:

1. FASA genera su propio certificado como autoridad raíz.
2. MVSA y CSSA reciben certificados firmados por FASA.

Jerarquía de autoridades de certificación que se ha desplegado

La jerarquía de certificación consiste en:asdfas

1. **Autoridad Raíz**: FASA (Fernando Alonso Shopping Authority) actúa como la autoridad raíz, creando y firmando su propio certificado.
2. **Autoridades Intermedias**: MVSA (Max Verstappen Shopping Authority) y CSSA (Carlos Saniz Shopping Authority) son autoridades intermedias cuyas claves públicas están certificadas por la autoridad raíz (FASA).
3. **Certificados finales / Certificados de los clientes**: Certificados para usuarios específicos, firmados por las autoridades intermedias correspondientes, dependiendo de la región (CSSA para España y MVSA para Países Bajos).

¿Por qué esta configuración y no otra?

Hemos elegido esta configuración porque ofrece escalabilidad y delegación, haciendo que la autoridad raíz (FASA) pueda delegar la emisión de certificados a autoridades intermedias (CSSA y MVSA).

También ofrece seguridad, usando la clave privada de la raíz que puede mantenerse segura (Por razones de la práctica, nos hemos tomado la licencia de guardar las claves públicas y privadas de las autoridades y de los usuarios en ficheros accesibles dado que no tenemos manera de guardar las claves privadas en directorios seguros como podría ser el archivo local del servidor de la entidad.) y usarse con menor frecuencia, mientras que las autoridades intermedias manejan la emisión de certificados “diariamente”.

A su vez, nos ofrece compatibilidad regional, CSSA se usa para certificados en España, y MVSA se utiliza para Países Bajos, reflejando una segmentación geográfica eficiente. En nuestro caso sólo hemos creado estos dos, ya que la aplicación se utilizaría imaginariamente en estos dos países, si se quisiera ampliar a más países se implementarían entidades certificadoras para cada país.

Esta jerarquía sigue las mejores prácticas de PKI, ya que separa claramente las responsabilidades y mejora la seguridad general del sistema.

Implementación

1. **Generación de claves y certificados:**
    a. **Claves privadas y públicas**: Se generan utilizando el módulo cryptography.hazmat.primitives.asymmetric.rsa.
    b. **Certificados**: Se crean y firman con cryptography.x509.CertificateBuilder.
2. **Estructura jerárquica:**
    a. **FASA** actúa como autoridad raíz (Root CA).
    b. **CSSA y MVSA** son autoridades intermedias firmadas por FASA.
    c. Los usuarios finales obtienen certificados firmados por las autoridades intermedias (CSSA o MVSA), dependiendo de su región.
3. **Almacenamiento:**
    a. Las claves privadas y públicas se almacenan en archivos .pem.(Como explicamos anteriormente somo conscientes de que las claves deberían guardarse en lugares seguros de manera que sólo el dueño de la clave pueda acceder a la clave privada,

pero nos hemos tomado la licencia para guardarlo en el mismo directorio por razones de la práctica)

b. Los certificados se guardan en directorios específicos organizados por autoridad.

4. **Carga de certificados existentes:**
   a. Si los certificados ya están generados, el sistema los carga desde los archivos en lugar de regenerarlos.

Uso de los certificados

Los certificados se utilizan en el momento de iniciar una conversación entre dos usuarios, donde el usuario que la empieza envía su clave simétrica (encriptada) junto con la cadena de certificados, para que, seguidamente, el que recibe esta información compruebe los certificados, viendo que la entidad raíz es la misma que la suya y por consecuencia puede confiar en el usuario que envió los certificados y en su entidad certificadora.

Los certificados se utilizan en una conversación entre dos usuarios para garantizar seguridad, autenticidad e integridad durante el intercambio de datos.

Autenticidad: El certificado asegura al receptor que la clave simétrica proviene de una entidad confiable. La cadena de certificados permite verificar la identidad del remitente a través de una jerarquía de confianza (Autoridad Raíz → Autoridades Intermedias → Certificado del usuario).

Integridad: Los certificados garantizan que cualquier intento de manipulación rompe la relación entre el contenido del certificado y la firma digital de dicho contenido, lo que invalida automáticamente esta última.

Complejidad

Nuestro código es bastante simple de usar, y la navegación es sencilla. El problema podría existir si no escribes las cosas tal cual se te pide, ya que solo detecta los valores de las cosas si se escriben bien. El diseño del código se ha intentado encapsular lo máximo posible, extrayendo todos los métodos de encriptación y desencriptación al servidor.
Nuestro programa implementa una especie de “Wallapop”, y permite intercambiar mensajes entre distintos usuarios que ofrecen productos al mercado. Dentro de nuestro programa se define el esquema de certificados que se va actualizando conforme más usuarios se registran, permitiendo obtener la cadena de certificados para cualquier usuario en cualquier momento.

Mejoras

Hemos implementado como mejoras el uso de una “base de datos” de forma local, utilizando jsons que son encriptados y solo accesibles por el “servidor”, así se garantiza una protección de la información.

También se produce validación de los datos que introduce el usuario, verificando que lo que dice está bien.