Protecting legal documents

📚 Problemática:

El director ejecutivo de un pequeño despacho de abogados desea que documentos legales como testamentos, fideicomisos, poderes notariales y diferentes contratos puedan realizarse de manera digital. En particular, busca que el proceso de firma pueda ser digitalizado. Un aspecto importante es que la mayoría de los documentos mencionados deben ser firmados por varias personas. Además, debe ser posible verificar la firma de dichos documentos. Adicionalmente, el director ejecutivo quiere mantener ciertos documentos en confidencialidad. En este escenario, cada caso legal se asigna a un equipo de abogados, por lo que únicamente los miembros del equipo deben poder abrir estos documentos. Ayuda al director ejecutivo de este despacho a desarrollar un pequeño sistema utilizando criptografía, para firmar digitalmente documentos y mantenerlos confidenciales.

💡 Solución propuesta:

Sistema web (front y back, servidor y clientes no locales) con tres tipos de usuarios.

Los usuarios ya deben estar asignados a su equipo.

1. Usuarios predefinidos junto con su contraseña, ordenados por caso legal:

⚖️ **Pensión alimenticia**	⚖️ **Divorcio**	⚖️ **Asunto hipotecario**
• Ramírez | mar789	• Pérez | mar456	• Castro | mar123
• Hidalgo | daniel789	• Cruz | daniel456	• Estrada | daniel123
• Boloñesa | avril789	• Perejil | avril456	• Mejía | avril123

2. Funciones y requerimientos de los usuarios:

👤 Abogado: Boloñesa, Cruz, Castro

- Genera par de claves RSA.
- Genera llave AES para cada documento.
- Sube/Cifra documentos.
- Cifra llave AES para cada miembro del equipo.
- Firma el contrato con el cliente.
- Firma otros documentos (demanda, escrito inicial).

👤 Cliente (persona física o moral): Ramírez e Hidalgo, Pérez y Perejil, Mejía

- Genera par de claves RSA.
- Descifra llave AES.

- Descifra documentos.
- Firma el contrato con el abogado.
- Firma otros documentos (demanda, escrito inicial).

👤 Otro (organismo de gobierno o notario): Estrada

- Genera par de claves RSA.
- Descifra documentos.
- Firma documentos (demanda, escrito inicial).

3. Funciones y requerimientos del sistema:

🖥️ Inicio de sesión

- Todos los usuarios se deben identificar con su nombre de usuario y contraseña.
- Todos los usuarios deben ejecutar el sistema web desde distintos dispositivos.
- El sistema web debe permitir la gestión de varios usuarios conectados al mismo tiempo.

🖥️ Generación de llaves

- Cada usuario deriva una única vez su par de claves RSA.
- Las claves privadas se descargan, en formato PEM.
- Las claves públicas de cada miembro del equipo se deben cargar automáticamente en el equipo, en formato PEM.
- El key wrapping de la llave AES se lleva a cabo con la clave pública RSA de cada miembro del equipo, utilizando OAEP para el padding.
- Cada llave AES cifrada debe cargarse automáticamente en el equipo, pero sólo el miembro deseado puede ver su llave cifrada.
- Cada miembro deseado del equipo puede descifrar la llave AES con su clave privada RSA, utilizando OAEP para el padding.
- La firma se lleva a cabo con la clave privada RSA del firmante, utilizando PSS para el padding.
- La verificación de las firmas se lleva a cabo con las llaves públicas RSA de cada firmante, utilizando PSS para el padding.

🖥️ Carga de documentos

- Sólo se pueden subir archivos PDF.
- No se pueden cargar automáticamente en el equipo hasta que se cifren.
- Los documentos se deben cifrar y descifrar con el cifrador de bloques AES y el modo de operación CBC.
- Una vez cifrado el documento, todos los miembros del equipo deben poder verlo desde su dispositivo.

- Una vez descifrado el documento, se puede visualizar, descargar y firmar, dependiendo el tipo de usuario.

🖥️ Firma y verificación de firmas de documentos

- Sólo se pueden firmar los documentos una vez que se descifren.
- Las firmas de cada firmante se deben cargar automáticamente en el equipo.
- Cualquier usuario puede verificar las firmas.
- Las firmas se verifican con el documento, la firma de cada firmante y la clave pública RSA de cada firmante, utilizando PSS para el padding.
- El hash del documento “original” y “descifrado” debe ser el mismo, es decir, no debe de alterarse durante el proceso de cifrado, carga y descifrado.

🗂️ Estructura de archivos:

```
proyecto/
├──── app.py                # Servidor Flask principal
├──── start.py              # Script de inicio
├──── uploads/              # Se crea automáticamente
├──── keys/                 # Se crea automáticamente
├──── documents/            # Se crea automáticamente
├──── signatures/           # Se crea automáticamente
├──── templates/
│    ├──── login.html
│    └──── dashboard.html
├──── static/
│    └──── dashboard.js
├──── sign/
│    ├──── __init__.py
│    ├──── digital_signer.py
│    ├──── signature_verifier.py
│    └──── key_generator.py
├──── cipher/
│    ├──── __init__.py
│    ├──── Cifrado_doc.py
│    ├──── Descifrado_doc.py
│    ├──── cifradollave.py
│    └──── decifradollave.py
```

🔐 Ejemplo caso legal de divorcio:

1. (Abogado) Cruz inicia sesión con su nombre de usuario y contraseña en el sistema web.
2. Le aparece su equipo con el nombre de “Divorcio”.

3. Le aparecen los nombres de Pérez y Perejil.
4. Cruz genera su par de llaves RSA (por única vez).
5. Sube un documento legal (Contrato de servicios profesionales) en el apartado del sistema.
6. Selecciona la opción para cifrar el documento.
7. Genera la llave AES CBC de manera segura y aleatoria.
8. El sistema la muestra el formato base64 para que Cruz la pueda copiar al portapapeles.
9. Cruz selecciona la opción para cifrar el documento, ingresa el documento y pega la llave AES desde el portapapeles.
10. El sistema genera el cifrado y lo carga automáticamente en el equipo.
11. El sistema muestra la opción para cifrar la llave AES con las claves públicas de cada miembro del equipo.
12. Cruz selecciona esta opción.
13. Cruz indica al sistema el nombre del archivo PEM con la clave pública de Pérez y de Perejil.
14. El sistema carga las llaves AES cifradas en el equipo para que cada miembro identifique la suya y pueda descifrarla.
15. Cruz firma el documento.
16. El sistema carga su firma automáticamente en el equipo para que todos en el equipo la puedan ver.
17. Cruz cierra su sesión.
18. (Cliente) Perejil inicia sesión con su nombre de usuario y contraseña en el sistema web.
19. Le aparece su equipo con el nombre de “Divorcio”.
20. Le aparecen los nombres de Pérez y Cruz.
21. Perejil genera su par de llaves RSA (por única vez).
22. Ve que hay un documento (Contrato) por descifrar.
23. Ve que hay una llave cifrada con su nombre.
24. Perejil descifra la llave AES cifrada con su clave RSA privada.
25. Descifra el documento (Contrato) con la llave AES descifrada.
26. Perejil firma el documento.
27. El sistema carga su firma automáticamente en el equipo para que todos en el equipo la puedan ver.
28. Perejil cierra su sesión.
29. Se repiten los pasos 18 a 28 para Pérez.
30. Se repiten los pasos 1 a 29 para el documento (Demanda) del divorcio.