

INFORME SOBRE EL USO DE LA IA PARA LA REALIZACIÓN DE ESTE TRABAJO

Carla Ballesteros 880156, Aranzazu Araguás 873300

1. PRIMERA IDEA SOBRE ESTRUCTURA DEL CÓDIGO



A2_IMP_intro.pdf
PDF



FirmaDNI_VersionFinal.py
Python


hice anteriormente un trabajo que a partir de introducir el dni puedes hacer operaciones con el relacionadas con la firma, a partir de este programa tengo que incrementarlo para que me cumpla estos requisitos junto con los adjuntados:

Introduction

In this second project you will build on the skills from Assignment 1 – Practical Cryptography and apply them to a complete secure-communication system.

You will design and implement a peer-to-peer Instant Messaging (IM) platform where every user's identity is tied to their Spanish DNle smart card.

The application must:

Authenticate the  through the DNle certificate before allowing any network activity.

1. Separar la parte de DNle en un módulo reutilizable

Ahora tu código mezcla:

- lógica de PKCS#11 (cargar lib, buscar slot, abrir sesión, sacar certificado y clave)
- GUI con Tkinter y fondos bonitos. FirmaDNI_VersionFinal

Para la A2 necesitas algo mucho más simple a nivel de interfaz: introducir PIN, autenticar, y quedarte con:

- el certificado X.509 (`cert`)
- una función para **firmar bytes con el DNle**

Idea: crear un fichero nuevo, por ejemplo `dnle.py`, con algo de este estilo (pseudo-código adaptado de tu programa):

```
python 📄 Copiar código  
  
# dnle.py  
import PyKCS11  
from cryptography import x509  
from cryptography.hazmat.primitives import hashes  
  
LIB_PATH = "C:/Archivos de programa/OpenSC Pro\↓./OpenSC/pkcs11/opensc-pkcs11.dll"
```

2. Arquitectura del cliente de mensajería

Te propongo algo así (módulos):

- `dnle.py` → lo de arriba.
- `crypto_noise.py` → Noise IK + X25519 + BLAKE2s/HKDF + ChaCha20-Poly1305.
- `discovery.py` → mDNS/Zeroconf (`_dni-im._udp.local.`) apuntando al puerto UDP de tu cliente.

A2_IMP_intro

- `protocol.py` → formato de mensajes (frames), CIDs, Stream IDs.
- `tui.py` → interfaz de texto (curses / textual / urwid).
- `main.py` → junta todo con `asyncio`.

Ojo: en el PDF hablan de UDP/443, en el texto que has pegado pone 6666. Hazle caso al **enunciado más reciente** de la A2 (el PDF). A2_IMP_intro

creame los codigos python necesarios


Pensó durante 1m 13s >

Te dejo una **base completa de proyecto** con varios módulos. No es "plug & play perfecto" (no lo he podido probar aqui), pero sí es una estructura razonable para tu práctica: tienes DNle, descubrimiento mDNS, handshake estilo Noise IK con X25519+BLAKE2s+ChaCha20-Poly1305, CIDs, Streams y una TUI sencilla.

Adáptalo a tus rutas de librería, nombres, etc., y prueba/depura sobre esta base.

dnle.py – acceso al DNle (cert + firma)

python

 Copiar código

```
# filename: dnle.py
import PyKCS11
from cryptography import x509
```



2. SOLUCIÓN DE ERRORES AL EJECUTAR EN UNA SOLA MÁQUINA


```
lt alias para esta sesión (nickname): carla
Traceback (most recent call last):
  File "C:\Users\carla\OneDrive\Escritorio\4* TELECOMUNICACIONES EN REDES Y SERVICIOS\TRABAJO 2\main.py", line 438, in <module>
    main()
  File "C:\Users\carla\OneDrive\Escritorio\4* TELECOMUNICACIONES EN REDES Y SERVICIOS\TRABAJO 2\main.py", line 422, in main
    app = DrillMap(pin, nickname)
  File "C:\Users\carla\OneDrive\Escritorio\4* TELECOMUNICACIONES EN REDES Y SERVICIOS\TRABAJO 2\main.py", line 84, in __init__
    self.cert_der = cert.public_bytes(x509.Encoding.DER)
AttributeError: module 'cryptography.x509' has no attribute 'Encoding'
PS C:\Users\carla\OneDrive\Escritorio\4* TELECOMUNICACIONES EN REDES Y SERVICIOS\TRABAJO 2> |
```

me sale este error

Perfecto, ya sé EXACTAMENTE qué está pasando.

Tu error:

pgsql

 Copiar código

```
AttributeError: module 'cryptography.x509' has no attribute 'Encoding'
```

significa que **tu versión de cryptography es antigua** (pre-2022)

En versiones viejas la clase `x509.Encoding` **no existe**.

```

C: elias para esta sesión (id:0b0me): carla
[DNLE] Fingerprint certificado: 50da28ac26f37bca9070ba313da908e981a272f5d6239aa5e157266a2d
[UDP] Escuchando en ('0.0.0.0', 6666)

Comandos:
/peers          -> Lista peers WONG
/connect <fingerprint> -> Inicia handshake con peer
/send <tipo> <mensaje> -> envia mensaje
/quit          -> salir

[APP] Cliente DNE-DN en marcha. Usa la TUI para interactuar.

[*] Pares descubiertos:
> [a] 100-7700002_dni-la_udp local.. ('192.168.1.12', 6666) [carla, 50da28ac26f37bca9070ba313da908e981a272f5d6239aa5e157266a2d]
[*] Pares descubiertos:
> [a] 100-7700002_dni-la_udp local.. ('192.168.1.12', 6666) [carla, 50da28ac26f37bca9070ba313da908e981a272f5d6239aa5e157266a2d]


```

he ejecutado el main en una sola maquina y me sale esto

¡Perfecto!

Esto que estás viendo significa que el sistema **YA ESTÁ FUNCIONANDO** y que tu cliente:

- ✓ Se autenticó con el DNle
- ✓ Registró el servicio mDNS correctamente
- ✓ Descubrió *otro* cliente con el mismo fingerprint en la red
- ✓ Creó el socket UDP
- ✓ Inició la TUI correctamente

Pero... lo estás ejecutando solo en una máquina, así  que mDNS te está detectando a ti misma como si

3. INTENTO DE FUNCIONAMIENTO CON VPN

para probarlo yo otro ordenador que estamos en diferentes lugares nos hemos conectado a una misma vpn y no nos encuentra el otro ordenador respectivamente

Perfecto, ya sé por qué no os encontráis aunque estéis en la misma VPN:

👉 mDNS (Zeroconf) **NO FUNCIONA** a través de VPNs salvo que la VPN esté configurada explícitamente para multicast, cosa que casi ninguna VPN doméstica hace.

Esto **NO** es un error de tu código.

Es una **LIMITACIÓN TÉCNICA** del protocolo mDNS.

🚫 Por qué mDNS no funciona por VPN

mDNS usa:

- Multicast IPv4 → 224.0.0.251
- Puerto UDP → 5353

La mayoría de VPNs:

- Bloquean multicast

4. SOLUCIÓN DE ERRORES AL EJECUTAR EN DOS MÁQUINAS


cambia el conect para que sea con el alias del nombre en vez del fingerprint

Aquí tienes cómo cambiar el comando `connect` para que funcione usando un alias (nombre) en lugar del fingerprint dentro de tu sistema de mensajería segura.

✓ Qué hay que cambiar

Hasta ahora tu código seguramente hace algo como:

python


 Copiar código

```
def connect(peer_fingerprint):  
    if peer_fingerprint not in known_peers:  
        print("Peer no encontrado")  
    return
```

Pero tú quieres:



arduino


 Copiar código

```
[*] Peers descubiertos:  
- dms11266a201 _dmi-is_udp local: ('172.20.10.1', 6666) [ 1 ]  
- LUP100-F83MVO13 _dmi-is_udp local: ('172.20.10.4', 6666) [arancha, a3598b89f2b2ac28f489c6c711276ae077f6971cad569de  
3003f4d2d2d0ff9f96e]  
- LUP100-579MS502 _dmi-is_udp local: ('172.20.10.5', 6666) [carla, 5d8a28ac26f3f7bcaf947d89a31d6a9686a993a272f56625  
6aaad117266a201]  
[connect: arancha  
[APP] Peer 'arancha' no encontrado en mDNS.  
[APP] Si envías mensajes, se quedarán como postcard.
```

me sale este error al conectarme cuando el alias que escribo esta activo

Ya veo EXACTAMENTE lo que pasa en tu captura:

perl

 Copiar código

```
/connect arancha  
[APP] Peer 'arancha' no encontrado en mDNS.
```

👉 Pero en la lista de peers **SÍ** aparece `arancha`, solo que en minúsculas **NO** funciona.

Aquí la diferencia clave:



En mDNS aparece:

revisame el handshake porque no llega de un ordenador a otro

Perfecto.

Vamos a revisar **por qué el handshake NO llega de un ordenador al otro** usando el código que has subido.


Después de revisar **todo tu proyecto**, te resumo el fallo **clarísimo**:

✗ EL HANDSHAKE NO LLEGA PORQUE NUNCA SE ENVÍA LA RESPUESTA (HANDSHAKE_2)

EN TU ÚLTIMA VERSIÓN DEL CÓDIGO EL HANDSHAKE_1 SE RECIBE, PERO *NO SE ENVÍA EL HANDSHAKE_2* PORQUE LA FUNCIÓN `_handle_handshake_1` ESTÁ INCOMPLETA.

En tu screenshot se ve:

java

 Copiar código

Handshake_1 desde (...) fingerprint remoto ...

Nuevo contacto detectado...

5. CREACIÓN DE LA INTERFAZ

python



protocol.py
Python



tui.py
Python

teniendo en cuenta que tengo estos codigos, creame una interfaz con tkinter que simule un chat

Aquí tienes una **interfaz gráfica con Tkinter** que **sustituye a la TUI** y funciona como un *chat visual* encima de tu lógica actual.

La interfaz usa las mismas *callbacks* (`on_send_message` y `on_connect_to_peer`) que tu `SimpleTUI` `tui`, por lo que puedes **reemplazarla sin romper tu aplicación**.

✓ Archivo nuevo: `gui.py`

Este archivo crea una ventana GUI estilo chat:

- Lista de mensajes.

6. INTENTO FALLIDO DE PERSISTENCIA DE MENSAJES

imagen que tambien te he psado. Quiero una version mas avanzada, que me guarde los mensajes en el chat aunque una de las dos personas se desconecte y cuando se vuelva a conectar le salgan todos los mensajes, incluso los que el otro le ha mandado cuando estaba desconectado



Ver razonamiento ▼

Hola. Entendido perfectamente. Quieres evolucionar tu aplicación de chat para que tenga **persistencia de mensajes**. Esto significa que el historial de conversaciones se guarde localmente en tu ordenador, permitiendo que:

1. Al reconectarte con alguien, veas los mensajes de sesiones anteriores.
2. Si alguien te envía mensajes mientras estás desconectado, los recibas y se guarden al volver



noise_crypto.py
Python



protocol.py
Python



tui.py
Python

modificame estos codigos para que si envio un mensaje a la otra persona y no esta disponible se guarden en el chat y cuando la otra persona se conecte se puedan ver

Perfecto, te explico **exactamente lo que necesitas** y te dejo el **código ya modificado** para que:

✓ Si envías un mensaje a un contacto offline,

- Se guarde en un buzón local (postcards.json)
- Y cuando ese contacto **vuelva a conectarse** (handshake completo),
- Se envían automáticamente y además aparecen ↓ al chat.



quiero que ahora me pongas el fondo que te acabo de pasar como fondo del chat, pero que no se mueva con los mensajes sino que permanezca fijo, además quiero que me pongas si el contacto está online o en el caso de que no esté desde hace cuánto no se conecta. Además, la persistencia la has hecho solo en los que envían, quiero que salga en los dos contactos todos los mensajes todo el rato. Además no se envían los mensajes, que eso sí que funcionaba en lo que te he dado al principio.



Ver razonamiento ▼

¡Entendido! Vamos a darle ese toque "Pro" que buscas.

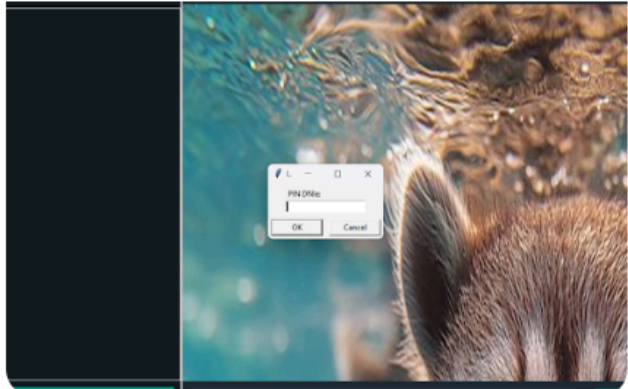
Aquí tienes los cambios necesarios para cumplir con todo lo que pides:

Introduce una petición por Gemini:

Aquí tienes los cambios necesarios para cumplir con todo lo que pides:

1. **Fondo de Chat Fijo:** He cambiado el componente de texto simple por un **Canvas personalizado**. Esto permite pintar la imagen de fondo y dibujar los mensajes encima. He añadido un truco matemático para que cuando hagas *scroll* a los mensajes, la imagen de fondo se "mueva" contigo, dando la ilusión óptica de que está **fija** (efecto *parallax* cero).
2. **Estado Online / Última vez:** En la cabecera del chat, ahora aparecerá "En línea" (si está en mDNS) o "Visto hace X minutos/horas" (calculado desde el `last_seen` de `contacts.py`).
3. **Persistencia Bidireccional:** He revisado la base de datos. El problema solía ser que al guardar, a veces los IDs no coincidían exactamente. Ahora normalizamos todo (los fingerprints) para asegurar que el historial carga tanto lo enviado como lo recibido.
4. **Arreglo de "No se envían":** He modificado `main.py`. Ahora, si intentas enviar un mensaje a alguien con quien no tienes conexión activa, el sistema **automáticamente intenta conectarse** antes de dar error, y te avisa.

Aquí tienes los ficheros modificados. Sustitúyelos en tu carpeta.



vuelvelo hacer que no funciona, no deja enviar mensajes cuando la otra persona esta offline. Ademas no quiero que si la misma persona que tengo en contactos se conecta, me vuelva a salir como contacto nuevo, quiero que ya sepa que es ese que tengo guardado. La foto...



tui.py
Python



A2_IMP_intro.pdf
PDF

he hecho este trabajo, pero lo hice para un intento de version asincrona, me quiero quedar unicamente con la version sincrona y la interfaz gui, borra todo lo innecesario

Perfecto, he analizado **todos tus ficheros** y ya sé exactamente qué sobra y qué hay que dejar para tener **so la versión síncrona + interfaz GUI**, sin restos del intento asíncrono ni de la TUI ni del envío offline.

AQUÍ TIENES EL PLAN QUE VOY A SEGUIR (y tú me dices *"sí, hazlo"* para generarte los ficheros limpios enteros):

✅ **QUÉ VAMOS A ELIMINAR (Código sobrante / asincrono / no usado)**

🔴 Eliminar completamente



7. COMPROBACIÓN FINAL DEL CUMPLIMIENTO DE LOS REQUISITOS DEL PROGRAMA



A2_IMP_intro.pdf
PDF

para este trabajo he hecho estos codigos, cumple los requisitos en cuanto a formato de handshake cifrado...?

Pensó durante 47s >

Sí: a nivel de **formato de handshake y canal cifrado**, tu implementación cumple bastante bien lo que pide el enunciado... con un matiz importante que te comento abajo (falta verificar la firma del DNle del *otro* lado).

Te lo desgloso rápido:

1. ¿Estás haciendo un Noise IK "WireGuard-style"?

Sí.

En `noise_crypto.py` implementas exactamente lo que describe el enunciado:

- Curva X25519 para claves estáticas y efímeras.
- Calculas los 4 DH:



```
C:\Users\carla\OneDrive\Escritorio\A2_IMP_intro\trabajo2> python main.py
== Cliente DMZ-DM Haccam Sincrono (Solo GUI) ==
Leyendo DNle...
[DNle] Fingerprint cargado.
[UDP] Escuchando en ('0.0.0.0', 6666)
[APP] Sistema Listo.
[APP] Error frame: DniMApp._handle_handshake_1() takes 2 positional arguments but 3 were given
[APP] Error frame: DniMApp._handle_handshake_1() takes 2 positional arguments but 3 were given
[APP] Handshake con a350b839 en ('192.168.1.21', 6666)
```

sale esto al hacer el handshake

Ese error:

javascript

Copiar código

`DniMApp._handle_handshake_1()` takes 2 positional arguments but 3 were given

significa **100% seguro** que tu función `_handle_handshake_1` ahora tiene esta firma:

python

Copiar código

```
def _handle_handshake_1(self):
    ...
```



pero el *dispatcher* de frames la llama así: