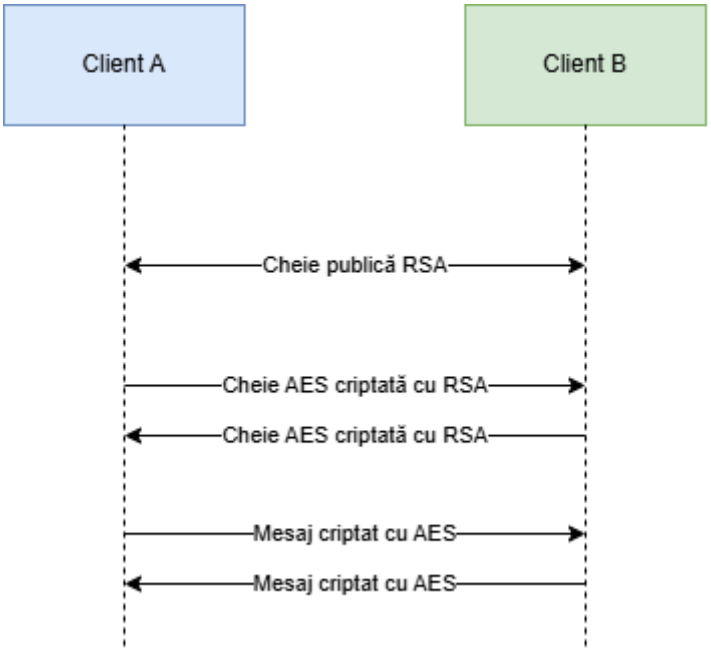
Proiect Securitate Informațională

Studeți: Carp Daniel-Cristian, Ciucanu Eric
Grupa: 1408A

1. Schema proiectului



2. Analiză

a) Algoritmi utilizați

AES (Advanced Encryption Standard) – implementat manual, cu toate etapele:

- SubBytes (folosind fișierul sbox)
- ShiftRows
- MixColumns
- AddRoundKey
- Extinderea cheii

RSA pentru negocierea cheilor AES. Include:

- Generare de perechi de chei (publică, privată)
- Criptare/decriptare a cheii AES (codificată ca string) cu RSA

b) Detalii de implementare

Modulul main.py conține clasa AES, care implementează complet algoritmul AES-128.

Exemplu de inițializare și criptare:

```
aes = AES()
aes.create_state("Hello World!")
aes.create_key("abcdefghijklmopq")
aes.cipher()
```

Operații implementate în ordine:

- create_state – convertirea textului într-o matrice de 4x4 octeți
- sub_bytes – substituția fiecărui byte folosind S-Box-ul
- shift_rows – rotirea liniilor 1–3
- mix_columns – amestecarea coloanelor folosind produsul Galois
- add_round_key – aplicarea cheii extinse

Extinderea cheii (key_expansion) este făcută conform specificației AES, generând 11 chei (1 inițială + 10 runde).

RSA

În fișierul rsa.py, cheia publică și cea privată sunt generate cu:

```
public_key, private_key = generate_keys(bits=512)
```

Funcțiile encrypt_string și decrypt_string criptează/decriptează mesaje ASCII convertite în întregi mari.

Negocierea cheii AES între clienți se face prin:

- Trimiterea cheilor publice în format JSON (json_key, format_json_key_to_tuple)
- Trimiterea cheii AES criptate cu cheia RSA a celuilalt

Client

În client.py, fiecare client poate asculta sau se poate conecta. După schimbul de chei RSA:

- Cheia AES este generată local și criptată cu RSA
- Este transmisă partenerului
- Comunicarea ulterioară este criptată AES bloc cu bloc (blocuri de 16 octeți)

Exemplu de trimitere AES:

```
for i in range(size16blocks):
    message_block = message_to_send[i * 16: (i + 1) * 16]
    other_AES.create_state(message_block)
    other_AES.create_key(recv_aes_key)
    other_AES.cipher()
    encrypted_block = hex_mat_to_ascii(other_AES.state)
```

Interfața de Utilizator (UI)

Interfața de utilizator a fost implementată folosind PyQt5, oferind o modalitate intuitivă pentru utilizatori de a interacționa cu sistemul de comunicare securizat. Clasa CommunicatorWindow gestionează interfața și logica de comunicare.

Funcționalități ale UI-ului:

- Conectare și Ascultare: Utilizatorii pot alege să se conecteze la un alt client sau să asculte pentru conexiuni în cadrul unui port specificat.

- Schimb de Chei: Schimbul de chei RSA și AES este gestionat automat la stabilirea conexiunii.
- Trimiterea și Primirea Mesajelor: Mesajele sunt criptate și decriptate automat folosind AES, iar utilizatorii pot vedea conversația în caseta de chat.

Metode Principale:

- listen: Pune clientul în mod de ascultare pentru conexiuni și gestionează schimbul de chei.
- connect: Conectează clientul la un alt client și gestionează schimbul de chei.
- receiver: Primește mesaje criptate, le decriptează și le afișează în caseta de chat.
- sender: Criptează mesajele și le trimite către celălalt client.

c) Dificultăți întâmpinate

- Manipularea blocurilor de 16 octeți: pentru AES este esențială alinierea mesajelor în blocuri de exact 16 octeți. A fost utilizat un padding PKCS#7.
- Conversia între formate: a fost necesară conversia între string-uri, byte array-uri și întregi mari pentru integrarea AES cu RSA.
- Gestionarea encodingului: pentru a păstra caracterele criptate și a evita caractere invalide, s-a folosit encoding-ul latin-1.
- Negocierea simetrică a cheii AES: a fost important ca ambele părți să trimită și să primească cheia AES criptată corect.