Securitate Informațională
Proiect

Proiect 3

Chiriac Alexandru 1409B
Lăduncă Ioan-Darian 1410A

github.com/DarianXD/Securitatea-Informatiei-Proiect

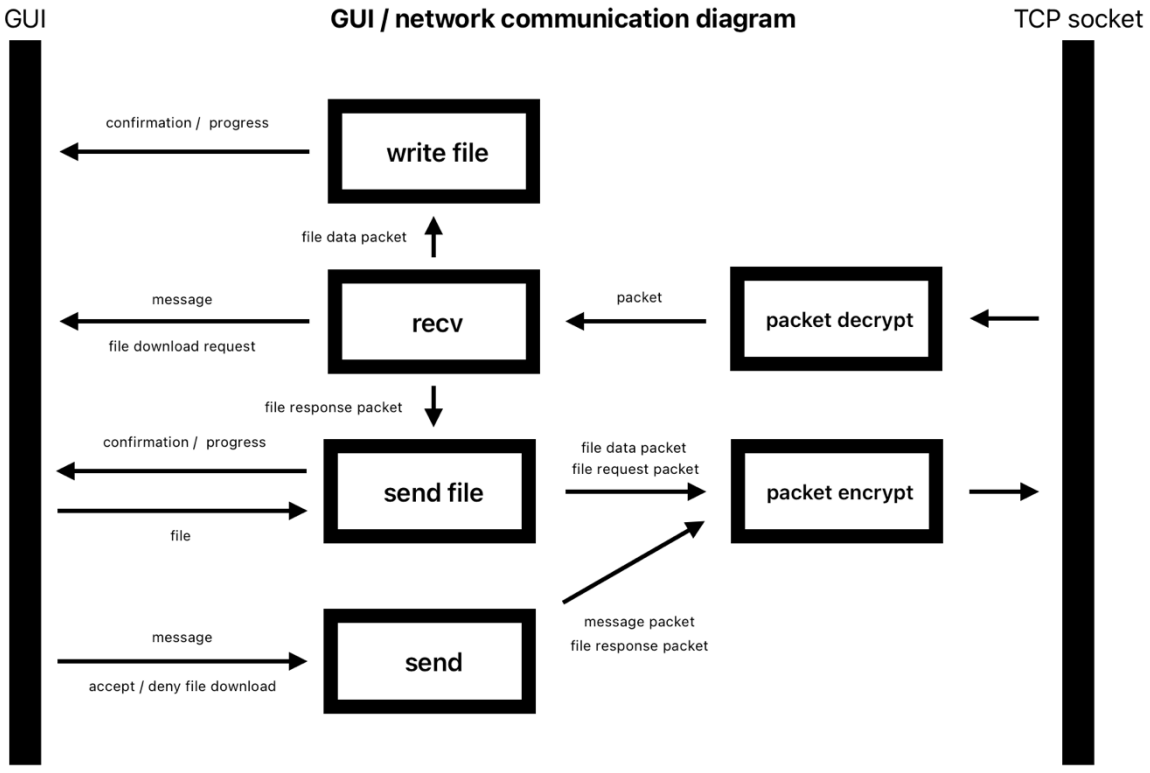
I. Descriere Proiect

Proiectul implementează un sistem de comunicare peer-to-peer pentru trimiterea de mesaje și fișiere criptate utilizând AES in modul ECB. Negocierea cheii de criptare se realizează utilizând protocolul de negociere Diffie-Hellman. Configurările necesare sunt salvate local într-un fișier, acestea nefiind transmise între clienți la realizarea conexiunii.

Limbajul de programare *C* a fost ales pentru implementarea algoritmului pentru a permite eficientizarea resurselor utilizate (timpul de execuție, memoria utilizată). Limbajul de programare *Python* și librăria *Tkinter* au fost alese deoarece permite dezvoltarea cu ușurință a aplicațiilor complexe rulând pe diferite sisteme de operare fără a modifica implementarea aplicației dar având posibilitatea de a utiliza cu ușurință librării de sistem.

II. Schema Proiectului

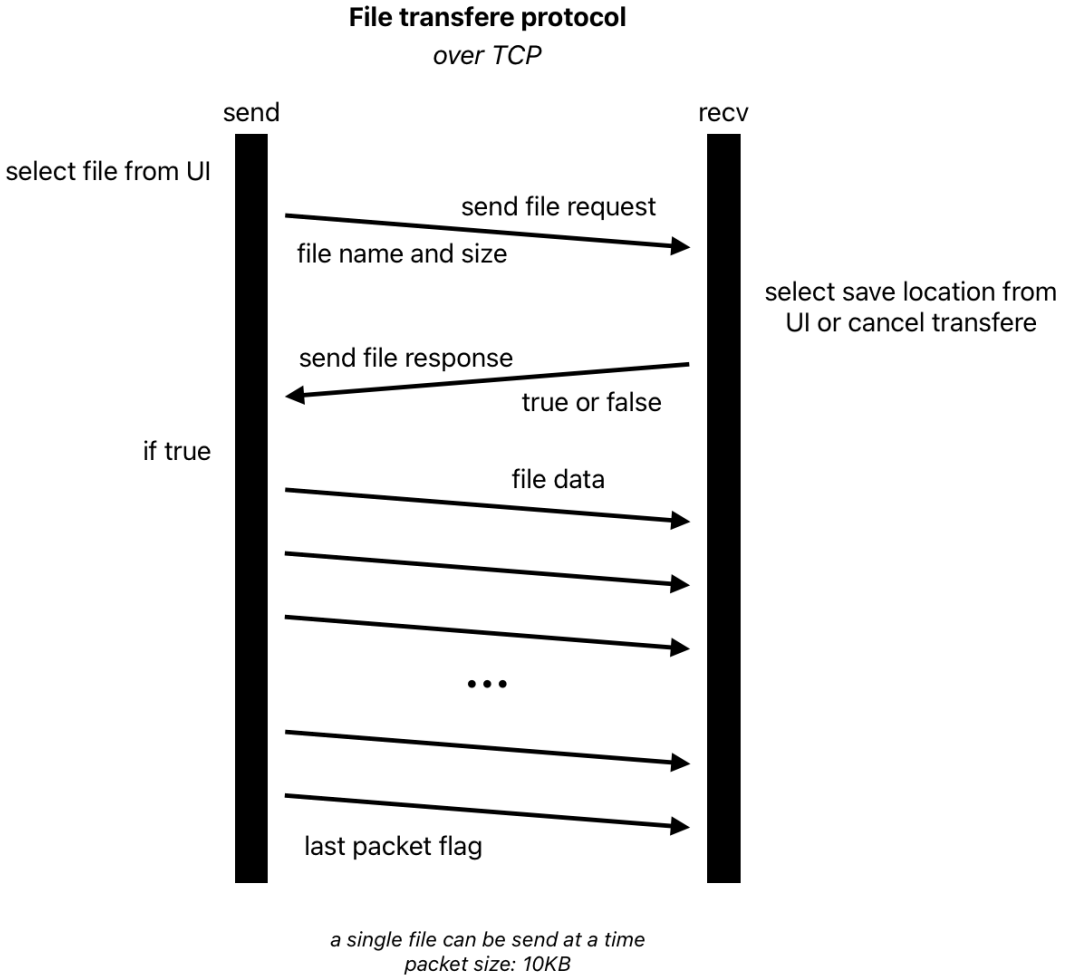
1. Diagramă de comunicație între componente



Componentele pentru *write file, send file, recv, send, GUI* ruleaze pe fire de execuție separate. Comunicarea între firele de execuție se realizează prin cozi de masaje pentru a transmite informații și prin flag-uri pentru a transmite statusul sau operația curentă de un anumit fir de execuție. Execuția acestor fire de execuție începe după stabilirea conexiunii și negocierea cheii de criptare.

Criptarea / serializarea și decriptarea / deserializare se realizează în firul de execuție care realizează operația corespunzătoare pachetului primit. Toate comunicatiile realzate între clienți după negocierea cheii sunt criptate.

2. Transmiterea fișierelor



3. Antetul pachetelor

File flag (1b)	Parameter (1b)	File request (1b)	File response (1b)	0 (4b)	Size (8B)	File Name Size (6B)

- Pachet mesaj: *file flag = 0, size = dimensiune mesaj (dimensiune date), date trimise = mesajul*
- Pachet fișier: *file flag = 1, file request = 0, file response = 0, size = dimensiune date, parameter = 0 (ultimul pachet) sau 1, date trimise = parte din fișier*
- Pachet cerere fișier: *file flag = 1, file request = 1, size = dimensiune fișier, file name size = dimensiune nume fișier (dimensiune date), date trimise = numele fișierului*
- Pachet răspuns fișier: *file flag = 1, file request = 0, file response = 1 size = 0, fără date trimise, parameter = 0 (descărcare acceptată) sau 1 (descărcare refuzată)*

Antetul și datele trimise sunt criptate separat folosind aceeași cheie de criptare. Dimensiune antetului plaintext este de 15 octeți. Dimensiunea antetului trimis este de 16 octeți. La citirea datelor se decriptează antetul, se citesc datele trimise în funcție de dimensiune specificată în antet, se decriptează și procesează datele primite.

III. Implementare

1. AES

Implementarea algoritmului de criptare *AES* este realizată în limbajul de programare *C*. Acesta este compilat sub forma unei librării dinamice *.so (Linux)* / *.dylib (Darwin)* utilizând compilatorul *Clang*. Acesta poate fi compilată sub forma unui *.DLL (Windows)* cu mici modificări în funcție de compilatorul folosit. Padding-ul utilizat prentru algoritm este PKCS7. Criptare se realizează în modul ECB.

Funcțiile expuse de librărie:

```
size_t AES_ECB(
    AES alg, const AES_key expanded_key, const byte *input,
    byte *output, size_t inputSize, size_t outputSize,
    bool decrypt
);
```

Criptarea sau decriptarea datelor

AES alg → algoritmul utilizat *(AES_128, AES_192, AES_256)*
const AES_key expanded_key → cheia returnată de funcția *AES_expand_key*
const byte *input → datele care vor fi criptate / decriptate
byte *output → zona de memorie unde va fi scris rezultatul
size_t inputSize → dimensiunea datelor de intrare
size_t outputSize → dimmensiunea zonei de memorie a rezultatului (între *inputSize* și *inputSize + 16* în funcție de padding)
bool decrypt → *0 = criptare, 1 = decriptare*
size_t return → *0 (eroare)* sau *dimensiunea datelor rezultate*

```
AES_key AES_expand_key(AES alg, const word *key);
```

Calculează cheia utilizată de funcția *AES_ECB*

AES alg → algoritmul utilizat *(AES_128, AES_192, AES_256)*
const word * key → cheia de criptare / decriptare
return AES_key → *0 (eroare)* sau cheia utilizată de funcția *AES_ECB*

```
void AES_free_key(AES_key *key);
```

Eliberează memoria alocată de *AES_expand_key*

AES_key *key → adresa de memorie a cheii returnată de funcția *AES_expand_key*

2. GUI

Interfața grafică este realizată în *Python* utilizând *Tkinter*. Comunicarea între utilizatori se este realizată prin TCP. Mesajele și fișierele trimise sunt criptate utilizând librăria AES implementată. Cheia de criptare și decriptare este generată pentru fiecare conexiune utilizând algoritmul Diffie-Hellman. Parametrii necesari negocierii *p* și *g* sunt stocați într-un fișier alături de algoritmul *AES* utilizat.

Trimiterea mesajelor și a fișierelor se realizează printr-o singură conexiune. Pentru a permite trimiterea mesajelor în timpul transferului de fișiere acestea sunt trimise în pachete cu dimensiunea de 10KB. Pentru a reduce numărul de citiri și scrieri pe disk fișierele sunt citite și scrise în blocuri de 250MB.

Mesajele și fișierele sunt criptare / decriptate în timpul trimiterii / primirii. Viteza de transmitere a fișierelor este de ~10MB/minut (~165KB/s). Înainte de a primi conținutul fișierului, clientul primește numele și dimensiunea acestuia și este întrebat dacă dorește să primească fișierul, locația în care să se salveze și numele cu care să se salveze utilizând meiurile oferite de sistemul de operare pe care rulează aplicația.

La apariția unei erori de criptare, decriptare sau de comunicație între clienți sau închiderea unui client conexiunea este închisă și clientul are opțiunea de a salva conținutul convorbirii.

Pentru rularea aplicației trebuie specificat în fișierul AES.py locația librăriei AES, aceasta fiind compilată în funcție de sistemul de operare și arhitectura acestuia.