Uma imagem com texto, Tipo de letra, captura de ecrã, Gráficos

Descrição gerada automaticamente

**Faculdade de Ciências da Universidade de Lisboa**

**Departamento de Informática**

**Mestrado em Engenharia Informática**

Relatório

**Privacidade e Segurança de Dados**

***Projeto (Primeira Parte)***

**Rodrigo Craveiro Rodrigues (Nº64370)**

**Diogo Serrano Sargaço (Nº58252)**

**André Filipe Diniz Belo (Nº58211)**

Professor: **Doutor** **Bernardo Ferreira**

1º Semestre Letivo 2024/2025

**outubro 2024**

Índice

[1. Introdução 3](#_Toc179578073)

[2. Funcionalidades Desenvolvidas 3](#_Toc179578074)

[2.1 Certificados Auto-assinados 3](#_Toc179578075)

[2.2 Troca de Chaves Segura 3](#_Toc179578076)

[2.3 Criptografia AES para Mensagens 3](#_Toc179578077)

[2.4Modo GCM 3](#_Toc179578078)

[2.5 Interface Gráfica e Gerenciamento de Conexões 3](#_Toc179578079)

[2.6 Manutenção de Histórico de Conversas 4](#_Toc179578080)

[3. Análise das Garantias de Segurança 4](#_Toc179578081)

[3.1 Confidencialidade 4](#_Toc179578082)

[3.2 Integridade e Autenticidade 4](#_Toc179578083)

[3.3 Autenticação de *Peer*s 4](#_Toc179578084)

[3.4 Resistência a Ataques MitM 4](#_Toc179578085)

[3.5 Armazenamento Seguro de Chaves e Certificados 4](#_Toc179578086)

[4. Considerações Finais 5](#_Toc179578087)

[5. Referências 5](#_Toc179578088)

[1. Socket 5](#_Toc179578089)

[2. Threading 5](#_Toc179578090)

[3. Tkinter 5](#_Toc179578091)

[4. Cryptography 6](#_Toc179578092)

[5. JSON 6](#_Toc179578093)

[6. OS e SYS 6](#_Toc179578094)

# 1. Introdução

O sistema desenvolvido é uma aplicação de chat *peer*-to-*peer* (P2P) que permite a comunicação direta entre utilizadores através de uma interface gráfica. O principal objetivo do sistema é garantir uma comunicação entre clientes de forma descentralizada, segura e autenticada, recorrendo a métodos como a criptografia assimétrica (RSA) e simétrica (AES) para proteger as mensagens e assegurar a integridade dos dados.

# 2. Funcionalidades Desenvolvidas

## 2.1 Certificados Auto-assinados

O sistema gera um par de chaves RSA (privada e pública) e um certificado auto-assinado para cada *peer*, permitindo a autenticação dos *peer*s e a criação de uma camada inicial de confiança.

## 2.2 Troca de Chaves Segura

Os *peer*s trocam os certificados no momento da conexão. Se o certificado de um *peer* estiver na lista de controlo de acesso (ACL), ele é considerado confiável. Caso contrário, o sistema adiciona automaticamente o certificado à ACL, promovendo uma confiança incremental.

## 2.3 Criptografia AES para Mensagens

Após a verificação de confiança, o sistema utiliza criptografia AES (256 bits) para cifrar as mensagens trocadas. A chave AES é encriptada com a chave pública do *peer* recetor, garantindo que apenas o *peer* correto possa desencriptá-la.

## 2.4Modo GCM

A criptografia AES é implementada no modo Galois/Counter Mode (GCM) para garantir não apenas a confidencialidade, mas também a integridade e autenticidade das mensagens, protegendo contra ataques de manipulação.

## 2.5 Interface Gráfica e Gerenciamento de Conexões

A aplicação permite aos utilizadores conectar-se a *peer*s inserindo o IP e a porta desejados, listando os *peer*s conectados e abrindo janelas de chat individuais para cada conexão ativa.

## 2.6 Manutenção de Histórico de Conversas

As mensagens trocadas são guardadas em ficheiros de texto, permitindo que o utilizador aceda ao histórico de conversas de cada *peer*.

# 3. Análise das Garantias de Segurança

## 3.1 Confidencialidade

A confidencialidade é assegurada através da utilização de criptografia AES de 256 bits para cifrar todas as mensagens enviadas entre os *peer*s. A chave AES é transmitida de forma segura utilizando criptografia assimétrica (RSA), garantindo que somente o destinatário legítimo possa decifrar e aceder ao conteúdo.

## 3.2 Integridade e Autenticidade

O modo GCM (*Galois/Counter Mode*) utilizado na implementação da criptografia AES não apenas cifra a mensagem, mas também gera um *tag* de autenticação que garante que qualquer alteração não autorizada nos dados seja detetada. Isso protege contra ataques de manipulação de mensagens.

## 3.3 Autenticação de *Peer*s

A aplicação utiliza certificados auto-assinados para autenticar os *peer*s e verificar sua identidade. Antes de estabelecer uma conexão segura, os certificados são comparados com a ACL local. Se um *peer* não estiver listado como confiável, ele é adicionado automaticamente, fortalecendo a segurança com base em um modelo de confiança incremental.

## 3.4 Resistência a Ataques MitM

A troca de chaves e a autenticação com certificados ajudam a mitigar ataques MitM (Man-in-the-Middle), pois um atacante precisaria comprometer as chaves privadas dos *peer*s para intercepta ou alterar a comunicação. Além disso, como as chaves AES são encriptadas com RSA e verificadas usando as chaves públicas dos certificados, a probabilidade de sucesso de um ataque MitM é minimizada.

## 3.5 Armazenamento Seguro de Chaves e Certificados

As chaves privadas e os certificados são armazenados localmente em formato PEM, sem qualquer forma de encriptação adicional. Para ambientes de produção, recomenda-se o uso de técnicas mais seguras, como o armazenamento das chaves em HSM (*Hardware Security Modules*) ou utilizando encriptação com senha.

# 4. Considerações Finais

O sistema P2P desenvolvido oferece um conjunto robusto de funcionalidades e mecanismos de segurança que garantem confidencialidade, integridade, e autenticação das comunicações entre os *peer*s. No entanto, algumas melhorias podem ser implementadas, como a integração de uma autoridade certificadora (CA) para substituir certificados autoassinados e a adição de autenticação baseada em senha para complementar a segurança do armazenamento das chaves privadas.

# 5. Referências

As principais bibliotecas e frameworks utilizados para a implementação desta primeira parte do projeto foram as seguintes.

## 1. **Socket**

A biblioteca usada para se criar sockets TCP/IP **(Python Standard Library)** para comunicação de rede entre os peers, é a biblioteca **socket**, esta que permite a criação de servidores e clientes TCP, essenciais para a estrutura P2P.

* **Documentação Oficial:** [Python Socket Library](https://docs.python.org/3/library/socket.html)

## 2. **Threading**

A biblioteca **threading** **(Python Standard Library)** permite que a aplicação execute múltiplas threads em simultâneo, facilitando a execução do servidor e da interface gráfica ao mesmo tempo.

* **Documentação Oficial:** [Python Threading Library](https://docs.python.org/3/library/threading.html)

## 3. **Tkinter**

A biblioteca **Tkinter** **(Python Standard Library)** é utilizado para criar a interface gráfica da aplicação, permitindo interação com os utilizadores de forma intuitiva e visual. A biblioteca é uma das ferramentas padrão para GUI em Python.

* **Documentação Oficial:** [Python Tkinter Library](https://docs.python.org/3/library/tkinter.html)

## 4. **Cryptography**

A biblioteca **cryptography** **(Third-Party Library)** é amplamente utilizada para implementar criptografia assimétrica (RSA) e simétrica (AES), além de manipular certificados x509 para autenticação de peers. É uma biblioteca moderna que provê métodos seguros para operações criptográficas.

* **Documentação Oficial:** Cryptography Library Documentation

### 5. **JSON**

A biblioteca **json** **(Python Standard Library)** é usada para manipular dados no formato JSON, especificamente para carregar e salvar a lista de peers confiáveis (ACL). Isso facilita a persistência de dados de forma estruturada e legível.

* **Documentação Oficial:** [Python JSON Library](https://docs.python.org/3/library/json.html)

### 6. **OS e SYS**

As bibliotecas **os** e **sys** **(Python Standard Library)** são usadas para manipular o sistema de arquivos e para operações do sistema operativo, como a criação de diretórios e manipulação de caminhos. São também utilizadas para encerrar a aplicação em caso de erros críticos.

* **Documentação Oficial:** [Python OS Library](https://docs.python.org/3/library/os.html), [Python SYS Library](https://docs.python.org/3/library/sys.html)