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](#_Toc179792257)

[2. Funcionalidades Desenvolvidas e Garantias de Segurança 3](#_Toc179792258)

[2.1 Certificados Auto-assinados e Autenticação de Peers 3](#_Toc179792259)

[2.2 Troca de Chaves Segura 3](#_Toc179792260)

[2.3 Criptografia AES para Mensagens 3](#_Toc179792261)

[2.4 Modo GCM e Garantia de Integridade e Autenticidade 4](#_Toc179792262)

[2.5 Interface Gráfica e Gestão de Conexões 4](#_Toc179792263)

[2.6 Manutenção de Histórico de Conversas 4](#_Toc179792264)

[2.6 Armazenamento Seguro de Chaves e Certificados 4](#_Toc179792265)

[2.6 Resistências a Ataques MitM 4](#_Toc179792266)

[3. Considerações Finais 4](#_Toc179792267)

[4. Referências 5](#_Toc179792268)

[4.1 Socket 5](#_Toc179792269)

[4.2 Threading 5](#_Toc179792270)

[4.3 Tkinter 5](#_Toc179792271)

[4.4 Cryptography 5](#_Toc179792272)

[4.5 JSON 5](#_Toc179792273)

[4.6 OS e SYS 6](#_Toc179792274)

# 1. Introdução

O sistema desenvolvido é uma aplicação python 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 troca de chaves Diffie-Hellman e criptografia simétrica (AES) para proteger as mensagens e assegurar a integridade dos dados.

# 2. Funcionalidades Desenvolvidas e Garantias de Segurança

## 2.1 Certificados Auto-assinados e Autenticação de Peers

O sistema utiliza um acordo de chaves Diffie-Hellman para cada peer, onde é implementado para estabelecer uma base segura para a autenticação dos peers antes de qualquer comunicação efetiva. No acordo de chaves Diffie-Hellman, cada peer gera sua própria chave privada e uma chave pública correspondente. As chaves públicas são então trocadas entre os peers antes de estabelecer uma conexão segura. O método Diffie-Hellman permite que dois peers estabeleçam um segredo compartilhado de forma segura, mesmo através de um canal inseguro. Este segredo pode ser usado para derivar chaves de sessão que cifram a comunicação subsequente.

Para além da segurança fornecida pelo Diffie-Hellman foi decidido implementar certificados para autentificação. Cada peer gera seu próprio certificado auto-assinado, onde o certificado inclui a chave pública do peer, que é uma chave Diffie-Hellman. Quando dois peers estabelecem uma conexão pela primeira vez, eles trocam seus certificados auto-assinados. Ao receber um certificado de outro peer, o sistema verifica sua validade. Uma vez que o certificado é verificado, se estabelece uma confiança entre os peers. Isso permite que comuniquem de forma segura, utilizando as chaves derivadas da troca de chaves Diffie-Hellman para cifrar a comunicação.

## 2.2 Troca de Chaves Segura

A segurança na troca de chaves utilizando o método Diffie-Hellman reside na capacidade de ambos os peers gerarem um segredo compartilhado que será exclusivo para cada sessão. tráfego. Após a troca das chaves públicas, cada peer utiliza sua chave privada e a chave pública do outro peer para gerar um segredo compartilhado. Este segredo é computacionalmente difícil de ser descoberto por terceiros, mesmo se interceptarem a troca de chaves públicas, devido à natureza do problema do logaritmo discreto em que o Diffie-Hellman se baseia.

O segredo compartilhado é então utilizado para derivar chaves de sessão que autenticam e criptografam as comunicações subsequentes. Este método não só fortalece a autenticação como também assegura que os dados trocados entre os peers não possam ser decifrados por entidades externas, reforçando a segurança contra ataques de intermediário MitM (Man-in-the-Middle).

## 2.3 Criptografia AES para Mensagens

Após a autenticação e troca de chaves, as mensagens entre os peers são cifradas usando criptografia AES de 256 bits. Isto garante que o conteúdo da comunicação esteja protegido contra acessos não autorizados, mantendo a confidencialidade dos dados. A chave AES é transmitida de forma segura utilizando o acordo de chaves Diffie-Hellman, assegurando que apenas o destinatário legítimo possa decifrar e aceder ao conteúdo das mensagens.

## 2.4 Modo GCM e Garantia de Integridade e Autenticidade

A criptografia AES é implementada no modo Galois/Counter Mode (GCM), que não só garante a confidencialidade das mensagens como também assegura a integridade e a autenticidade das mesmas. O modo GCM gera um *tag* de autenticação que detecta qualquer alteração não autorizada nos dados. Isso oferece uma proteção adicional contra ataques de manipulação, prevenindo que um atacante possa modificar ou forjar mensagens sem ser detetado.

## 2.5 Interface Gráfica e Gestão de Conexões

A aplicação oferece uma interface gráfica intuitiva, onde os utilizadores podem conectar-se a peers, inserindo o respetivo IP e porta. A interface permite listar os peers conectados e abrir janelas de chat individuais para cada conexão ativa, facilitando a gestão de várias conversas simultâneas. Esta camada gráfica é essencial para oferecer uma experiência de utilizador eficiente e amigável, simplificando o processo de conexão e comunicação entre os peers.

## 2.6 Manutenção de Histórico de Conversas

Para garantir que o utilizador possa aceder a conversas anteriores, o sistema armazena as mensagens trocadas em ficheiros de texto. Cada peer tem o seu próprio ficheiro de histórico, o que permite ao utilizador rever conversas passadas de forma fácil e organizada.

## 2.6 Armazenamento Seguro de Chaves e Certificados

As chaves privadas e os certificados são armazenados localmente em formato PEM.

## 2.6 Resistências a Ataques MitM

A arquitetura do sistema é projetada para minimizar a vulnerabilidade a ataques MitM. A autenticação baseada em certificados e a troca segura de chaves garantem que um atacante precise de comprometer as chaves privadas dos peers para intercepta ou modificar a comunicação. Como a chave AES é encriptada utilizando RSA, e a autenticidade é verificada através das chaves públicas dos certificados, o risco de sucesso de um ataque desse tipo é significativamente reduzido.

# 3. 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 auto-assinados, a adição de autenticação baseada em senha para complementar a segurança do armazenamento das chaves privadas, a encriptação dos ficheriros de histórico de conversa, e o armazenamento das chaves privadas e certificados em HSM (Hardware Security Modules).

# 4. Referências

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

## 4.1 **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)

## 4.2 **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)

## 4.3 **Tkinter**

A biblioteca **Tkinter** **(Python Standard Library)** é utilizado para criar a interface gráfica da aplicação, permitindo a interação entre 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)

## 4.4 **Cryptography**

A biblioteca cryptography (Third-Party Library) é utilizada para implementar a troca de chaves Diffie-Hellman, bem como criptografia simétrica (AES), além de manipular certificados para autenticação de peers. É uma biblioteca moderna que provê métodos seguros para operações criptográficas.

* **Documentação Oficial:** Cryptography Library Documentation

## 4.5 **JSON**

A biblioteca **json** **(Python Standard Library)** é usada para manipular dados no formato JSON, especificamente para carregar e guardar 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)

## 4.6 **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)