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

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 ao protocolo de troca de chaves de Diffie-Hellman com curvas elípticas, e a 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

É implementado para o estabelecimento de conexões de forma segura e com autenticação dos peers o protocolo de Diffie-Hellman com curvas elípticas (ECDH), que reside na capacidade de ambos os peers gerarem um segredo compartilhado que será exclusivo para cada sessão. 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. Desta forma, este protocolo método 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. Durante o handshake inicial, os peers trocam esses certificados, e embora que estes sejam auto-assinados, permite que cada peer tenha uma identidade única baseada na chave pública. 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.

Na aplicação, a geração dos certificados é realizada com recurso da biblioteca “*cryptography”*. Os certificados são armazenados no diretório *certificates* e descarregados quando a aplicação é iniciada. A troca de certificados ocorre nos métodos “*handle_new_connection”* e “*connect_to_peer”*, onde os peers enviam e recebem os certificados antes de proceder com a troca de chaves publicas.

2.2 Troca de Chaves Segura

Para a realização de troca de chaves de forma segura recorremos ao uso de certificados autoassinados por cada peer e ao uso do protocolo Diffie-Hellman com curvas elípticas (ECDH), que reside na capacidade de ambos os peers gerarem um segredo compartilhado que será exclusivo para cada sessão. Após a troca de certificados, os peers geram um par de chaves ECDH e trocam as chaves públicas correspondentes. Isso permite que ambos os peers calculem um segredo compartilhado que será utilizado para derivar a chave simétrica para criptografia das mensagens.

Na aplicação, é executada a geração das chaves ECDH pela função "*generate_key_pair"*, e a troca ocorre nos mesmos métodos mencionados anteriormente. O segredo compartilhado é calculado com recurso do método "***Exchange**"* da chave privada, passando a chave pública do peer.

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. Este modo oferece uma proteção adicional contra ataques de manipulação, prevenindo que um atacante possa modificar as mensagens sem ser detetado.

Nas funções de criptografia e descriptografia, o *nonce* e o *tag* de autenticação são manipulados, onde o valor de nonce é gerado aleatoriamente para cada mensagem, e o valor de tag é anexado no final do ciphertext. Desta forma, durante a descriptografia, o tag é verificado, garantindo a integridade da mensagem.

2.5 Interface Gráfica e Gestão de Conexões

A nossa aplicação oferece uma interface gráfica intuitiva, na qual foi desenvolvida com recurso da biblioteca "Tkinter". Nesta, os utilizadores podem conectar-se a peers, inserindo o respetivo IP e PORT, onde de seguida permite aos utilizadores observar uma lista de outros peers conectados com a possibilidade de ser selecionadas janelas de chat individuais para cada conexão ativa, facilitando a gestão das várias conversas de forma simultânea. A gestão de

conexões é realizada tanto pelo servidor (aceitando conexões entrantes) quanto pelo cliente (iniciando conexões). São utilizadas várias threads para permitir que haja múltiplas conexões simultaneamente, garantindo que a interface permaneça responsiva. 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, isto de uma forma simples, mantemos um histórico das conversas com cada peer, onde armazenamos as mensagens em arquivos de texto nomeados com o IP e a PORT do peer. Ao ser aberto uma janela de chat, o histórico é carregado e exibido para o utilizador.

As funções *save_chat_to_file* e *load_chat_from_file* são responsáveis por guardar e descarregar o histórico das conversas, o que permite ao utilizador ter acesso ao histórico mesmo após o reiniciar da aplicação.

2.7 Armazenamento Seguro de Chaves e Certificados

Para garantir que o utilizador possa aceder a conversas anteriores, as chaves privadas e os certificados são armazenados localmente no diretório "*certificates"*. As chaves privadas são salvas em formato PEM sem criptografia. Embora isso facilite o desenvolvimento, em um ambiente de produção, é recomendável proteger as chaves privadas com uma senha ou utilizar um armazenamento seguro.

No código atual, a função "*load_or_generate_certificate"* cuida de carregar ou gerar as chaves e certificados necessários, assegurando que cada peer tenha um par de chaves único.

2.8 Resistências a Ataques MitM

A arquitetura do sistema implementa medidas para resistir a ataques MitM, como o uso de ECDH para troca de chaves e AES-GCM para a criptografia das mensagens trocadas entre peers. No entanto, como os certificados são auto-assinados e não há verificação robusta da identidade dos peers, a aplicação ainda é vulnerável a ataques MitM.

Para fortalecer a resistência a ataques MitM, seria necessário implementar um mecanismo de verificação de certificados, como uma infraestrutura de chave pública (PKI) ou verificação de impressões digitais (fingerprints) dos certificados através de um canal seguro alternativo.

3. Referências

As principais bibliotecas e frameworks utilizados para a implementação, encontram-se no link. (Link:https://github.com/Dnasar0/PSD/blob/main/References1_PSD.txt)