Uma imagem com texto, Tipo de letra, design

Descrição gerada automaticamente

Segurança e Privacidade de Dados

Trabalho 2

Mestrado em Engenharia Informática – Engenharia de Dados

2023 / 2024

1190384 – André Teixeira

1190818 – Luís Pinto

1231925 – João Henriques



Resumo

O relatório aborda a implementação de uma Infraestrutura de Chave Pública (PKI) para garantir a segurança das comunicações e dos dados em repouso de uma empresa virtual, seguindo as melhores práticas de segurança. O problema central é a complexidade e a gestão desafiadora da criptografia em ambientes empresariais, o que pode levar à diminuição dos controles de segurança e ao aumento das ameaças aos sistemas de informação. A metodologia inclui a criação de um Certificado de Autoridade (CA) interno e externo, a definição de políticas de segurança, a implementação de autenticação por chave em uma aplicação protegida por um proxy reverso e o desenvolvimento de procedimentos para onboarding, revogação e recuperação de identidades.

**Palavras-chave:**

PKI, segurança, criptografia, certificados, CA, TLS, políticas de segurança, autenticação, revogação de identidade, recuperação de identidade, gestão de certificados, infraestrutura de chave pública, aplicação de autenticação.

ÍNDICE

[1. Introdução 4](#_Toc166442002)

[2. Implementação 5](#_Toc166442003)

[3. Conclusão 17](#_Toc166442004)

[Referências 18](#_Toc166442005)

ÍNDICE DE FIGURAS

[Figura 1 rootCA 5](#_Toc166443869)

[Figura 2 Imports do projeto 6](#_Toc166443870)

[Figura 3 Def issue\_certificate 7](#_Toc166443871)

[Figura 4 revoke\_certificate 7](#_Toc166443872)

[Figura 5 list\_certificates 8](#_Toc166443873)

[Figura 6 generate\_user\_certificate 9](#_Toc166443874)

[Figura 7 revoke\_user\_certificate 9](#_Toc166443875)

[Figura 8 recover\_user\_identity 10](#_Toc166443876)

[Figura 9 encrypt\_message 10](#_Toc166443877)

[Figura 10 decrypt\_message 10](#_Toc166443878)

[Figura 11 generate\_key\_pair 11](#_Toc166443879)

[Figura 12 save\_certificates 11](#_Toc166443880)

[Figura 13 load\_certificates 11](#_Toc166443881)

[Figura 14 issue\_ca\_certificate 12](#_Toc166443882)

[Figura 15 revoke\_ca\_certificate 12](#_Toc166443883)

[Figura 16 list\_ca\_certificates 13](#_Toc166443884)

[Figura 17 issue\_user\_certificate 13](#_Toc166443885)

[Figura 18 revoke\_user\_certificate 13](#_Toc166443886)

[Figura 19 list\_user\_certificates 14](#_Toc166443887)

[Figura 20 encrypt\_message 14](#_Toc166443888)

[Figura 21 decrypt\_message 15](#_Toc166443889)

# 1. Introdução

Este relatório foi realizado no âmbito da unidade curricular de Segurança e Privacidade de Dados, incluída no primeiro ano do mestrado em Engenharia Informática, ramo de Engenharia de Dados, do Instituto Superior de Engenharia do Porto.

O nosso objetivo principal foi implementar uma Infraestrutura de Chave Pública (PKI) seguindo as melhores práticas de segurança, garantindo a segurança das comunicações e dados em repouso. Isso inclui a criação de um Certificado de Autoridade (CA) interno e externo, emissão e gestão de certificados para utilizadores e serviços, implementação de políticas de segurança, autenticação por chave em vez de senha numa aplicação protegida por um proxy reverso e recuperação de identidades.

# 2. Implementação

Este comando gera uma chave privada para a Autoridade de Certificação com o nome do arquivo rootCA.key e um comprimento de chave de 4096 bits.

A black screen with white text

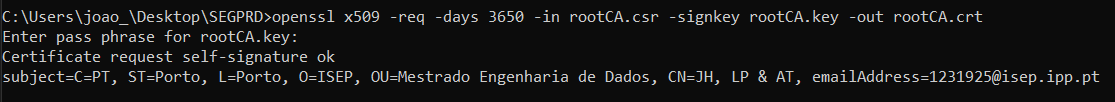
Description automatically generated

Figura 1 rootCA

A screenshot of a computer program

Description automatically generated

Pass: segprd



Irá ser explicado o processo de implementação através da explicação de *snipets* do código criado para o problema exposto.

Começamos por fazer os *imports* das bibliotecas *python* utilizadas para o projeto.

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

Descrição gerada automaticamente

Figura 2 Imports do projeto

Este código define uma classe “CertificateAuthority” que representa uma Autoridade Certificadora (CA). Uma CA é uma entidade confiável que emite certificados digitais para autenticar a identidade de entidades em um sistema de segurança, como em comunicações seguras pela internet.

* A classe “CertificateAuthority” possui um método “\_\_init\_\_ “que inicializa um dicionário(estrutura de dados que mapeia chaves para valores) vazio chamado certificates. Este dicionário será usado para armazenar os certificados emitidos pela autoridade certificadora.
* O método “issue\_certificate” é responsável por emitir um certificado para um determinado sujeito (identidade). Ele aceita dois argumentos: subject\_name (o nome do sujeito para quem o certificado está sendo emitido) e validity\_period (o período de validade do certificado).

No interior deste método, é chamada uma função generate\_key\_pair() para gerar um par de chaves público-privadas para o utilizador. No entanto, a implementação dessa função não está presente no código fornecido.

Depois de gerar o par de chaves, o método cria um certificado contendo informações sobre o user, a chave pública e o período de validade.

A chave pública é convertida num formato PEM (Privacy-Enhanced Mail) usando a biblioteca serialization do módulo cryptography.

O certificado é então armazenado no dicionário certificates, utilizando o nome do sujeito como chave.

Finalmente, o método retorna o certificado emitido.´

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

Descrição gerada automaticamente

Figura 3 Def issue\_certificate

Este método revoga o certificado associado ao nome do sujeito especificado. Ele verifica se o nome do sujeito está presente no dicionário de certificados da CA. Se estiver presente, o certificado é removido usando del

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

Descrição gerada automaticamente

Figura 4 revoke\_certificate

Este método simplesmente retorna todos os certificados emitidos pela CA, que são armazenados no dicionário “self.certificates”. Retorna o dicionário completo contendo todos os certificados.

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

Descrição gerada automaticamente

Figura 5 list\_certificates

O método “\_\_init\_\_” é o construtor da classe. Inicializa um dicionário vazio chamado user\_identities, que será usado para armazenar as identidades dos utilizadores, juntamente com os seus pares de chaves público-privadas.

* Já o método generate\_user\_certificate é usado para gerar um certificado para um utilizador específico. Ele aceita o nome de utilizador como entrada.
* Dentro deste método, um par de chaves RSA é gerado usando uma função chamada ”generate\_key\_pair()”. Depois de gerar o par de chaves, um certificado é criado contendo o nome de utilizador e a chave pública correspondente.

A chave pública é convertida para o formato PEM usando a biblioteca “serialization” do módulo cryptography.

O par de chaves (chave privada e pública) é armazenado no dicionário “user\_identities”, onde a chave é o nome de utilizador e o valor é uma tupla(semelhante a uma lista, mas com a distinção principal de que é imutável)contendo a chave privada e a chave pública.

Finalmente, o método retorna o certificado gerado para o utilizador.

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

Descrição gerada automaticamente

Figura 6 generate\_user\_certificate

Esta função faz parte da classe “IdentityProvider” e é responsável por revogar o certificado de um utilizador específico, identificado pelo nome de utilizador fornecido

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

Descrição gerada automaticamente

Figura 7 revoke\_user\_certificate

Esta função “recover\_user\_identity “ também faz parte da classe “IdentityProvider” e é usada para recuperar a identidade de um utilizador específico, identificado pelo nome de utilizador fornecido como argumento. É útil em casos de perda ou comprometimento da identidade do user.

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

Descrição gerada automaticamente

Figura 8 recover\_user\_identity

Este método é usado para criptografar uma mensagem usando a chave pública do destinatário. Ele aceita dois argumentos: a mensagem a ser criptografada e a chave pública do destinatário.

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

Descrição gerada automaticamente

Figura 9 encrypt\_message

Já por outro lado, este método é usado para descriptografar uma mensagem usando a chave privada do destinatário. Ele aceita dois argumentos: o texto cifrado e a chave privada do destinatário

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

Descrição gerada automaticamente

Figura 10 decrypt\_message

Esta função gera um par de chaves RSA, composto por uma chave privada e uma chave pública. Usa a biblioteca cryptography para gerar as chaves com um tamanho de chave de 2048 bits. A chave privada é retornada como “private\_key” e a chave pública é derivada da chave privada e retornada como “public\_key”.

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

Descrição gerada automaticamente

Figura 11 generate\_key\_pair

Esta função guarda os certificados emitidos pela CA num arquivo JSON chamado “certificates.json”

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

Descrição gerada automaticamente

Figura 12 save\_certificates

Esta função carrega os certificados emitidos pela CA a partir do arquivo JSON “certificates.json”

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

Descrição gerada automaticamente

Figura 13 load\_certificates

Este método é chamado quando o utilizador deseja emitir um certificado para a própria Autoridade Certificadora. Ele obtém o nome do sujeito (no caso, a Autoridade Certificadora) e o período de validade a partir da interface do user. Em seguida, chama o método “issue\_certificate” da instância ca (que é uma instância da classe CertificateAuthority) para emitir o certificado para a CA. Depois de emitir o certificado, chama a função “save\_certificates” para salvar os certificados atualizados (incluindo o novo certificado emitido para a CA) num arquivo JSON. Por fim, exibe uma mensagem informativa para o utilizador indicando que o certificado da CA foi emitido com sucesso.

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

Descrição gerada automaticamente

Figura 14 issue\_ca\_certificate

Este método é chamado quando o utilizador deseja revogar o certificado da Autoridade Certificadora. Ele obtém o nome do sujeito (novamente, a CA) da interface do user e, em seguida, chama o método “revoke\_certificate” da instância ca para revogar o certificado da CA. Depois de revogar o certificado, chama a função save\_certificates para salvar os certificados atualizados (com o certificado da CA revogado) num arquivo JSON.

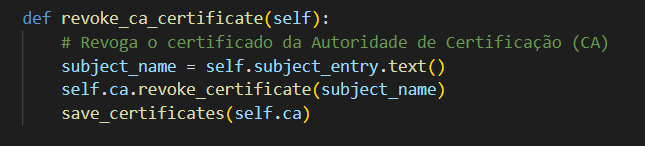


Figura 15 revoke\_ca\_certificate

Este método é chamado quando o utilizador deseja listar os certificados emitidos pela Autoridade Certificadora (CA). Ele obtém os certificados da CA chamando o método “list\_certificates” da instância ca (que é uma instância da classe CertificateAuthority).

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

Descrição gerada automaticamente

Figura 16 list\_ca\_certificates

Este método é chamado quando o utilizador deseja emitir um certificado para um user. Ele obtém o nome do utilizador da interface do user e, em seguida, chama o método “generate\_user\_certificate” da instância identity\_provider (que é uma instância da classe IdentityProvider) para gerar o certificado para o utilizador.

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

Descrição gerada automaticamente

Figura 17 issue\_user\_certificate

Este método é chamado quando o usuário deseja revogar o certificado de um utilizador.

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

Descrição gerada automaticamente

Figura 18 revoke\_user\_certificate

Este método é chamado quando o utilizador deseja listar os certificados emitidos para os utilizadores. Ele itera sobre o dicionário “user\_identities” da instância identity\_provider para obter todos os certificados dos usuários. Para cada utilizador, obtém a chave pública associada e a converte em uma string PEM

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

Descrição gerada automaticamente

Figura 19 list\_user\_certificates

Este método criptografa uma mensagem usando a chave pública do destinatário. Ele obtém o nome do destinatário e a mensagem da interface do utilizador. Em seguida, verifica se o destinatário está presente no dicionário “user\_identities” da instância identity\_provider, que contém as identidades dos utilizadores. Se o destinatário estiver presente, recupera sua chave pública e usa a instância da classe “EndToEndEncryption” para criptografar a mensagem usando a chave pública do destinatário.

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

Descrição gerada automaticamente

Figura 20 encrypt\_message

Por outro lado, este método descriptografa uma mensagem usando a chave privada do destinatário. Ele também obtém o nome do destinatário e o texto cifrado da interface do utilizador. Da mesma forma que no método “encrypt\_message”, verifica se o destinatário está presente no dicionário “user\_identities”. Se o destinatário estiver presente, recupera sua chave privada e usa a instância da classe “EndToEndEncryption” para descriptografar o texto cifrado usando a chave privada do destinatário

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

Descrição gerada automaticamente

Figura 21 decrypt\_message

# 3. Conclusão

A criação de um *chat* encriptado permitiu aprofundar os conhecimentos do grupo em relação aos vários termos de encriptação, nomeadamente, chaves públicas e privadas, protocolo *Tor*, certificados, entre outros.

Ao longo deste trabalho foi possível compreender as ameaças à informação e as estratégias utilizadas para as mitigar. A necessidade de proteger a privacidade e confidencialidade dos dados é cada vez mais importante dada a crescente quantidade de dados que o utilizador partilha nas diferentes plataformas.

Percebemos também que a segurança é uma jornada continua e que ferramentas como este *chat* continuarão a evoluir para enfrentar os novos desafios e ameaças que irão aparecer.

O trabalho foi dividido de forma justa e equilibrada de forma que todos os elementos do grupo tivessem impacto no desenvolvimento e que o resultado cumprisse os requisitos pedidos no enunciado.

# Referências

*MoodleISEP*. (n.d.). Moodle.isep.ipp.pt. Retrieved March 24, 2024, from <https://moodle.isep.ipp.pt/course/view.php?id=4744>

*ONION SERVICES | Tor Project | Tor Browser Manual*. (n.d.). Tb-Manual.torproject.org. <https://tb-manual.torproject.org/onion-services/>

*ngrok Platform Overview | ngrok documentation*. (n.d.). Ngrok.com. <https://ngrok.com/docs/>

Cruz, J. D. (2023, September 29). *Chaves Públicas e Privadas: Como Funcionam e Por Que São Importantes*. Medium. <https://medium.com/@jonathandacruz/chaves-p%C3%BAblicas-e-privadas-como-funcionam-e-por-que-s%C3%A3o-importantes-59a6da146592>

Software, C. W. I. (n.d.). *Dúvidas Terra*. Dúvidas Terra. Retrieved March 24, 2024, from <https://duvidas.terra.com.br/duvidas/570/o-que-e-criptografia-de-chaves-publica-e-privada>

GOLDSCHLAG, David; REED, Michael; SYVERSON, Paul. Onion Routing for Anonymous and Private Internet Connections. 1999 from: <https://www.onion-router.net/Publications/CACM-1999.pdf>

Sending Files Over the Internet with Python’s http.server and Ngrok: A File Transfer Hack for Restricted Environments from: <https://medium.com/@jonathan.hoffman91/sending-files-over-the-internet-with-pythons-http-server-14d5446f29b4>

Rede TOR e Onion Routing from: <https://www.dio.me/articles/rede-tor-e-union-routing>

‌

‌

‌