Encriptação e Decriptação de arquivos (Simétrica e Assimétrica)

QXD0099 - Desenvolvimento de Software para Persistência

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br

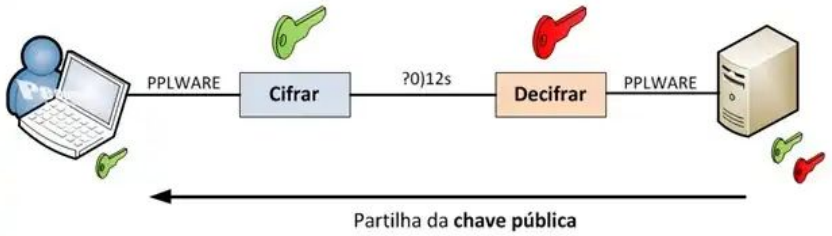
Agenda

- O que é criptografia?
- Tipos de criptografia
- Encriptação Simétrica
  - Principais algoritmos
  - Implementação prática
  - Aplicações práticas
- Encriptação Assimétrica
  - Principais algoritmos
  - Como funciona
  - Implementação prática
  - Aplicações práticas
- Comparação: Simétrica vs Assimétrica

O que é criptografia?

- Criptografia é um método de segurança de dados que utiliza algoritmos matemáticos para transformar informações legíveis em texto cifrado, ou seja, incompreensível para quem não tiver a chave correta.

- A chave é um conjunto de valores matemáticos que tanto o remetente como o destinatário concordam.



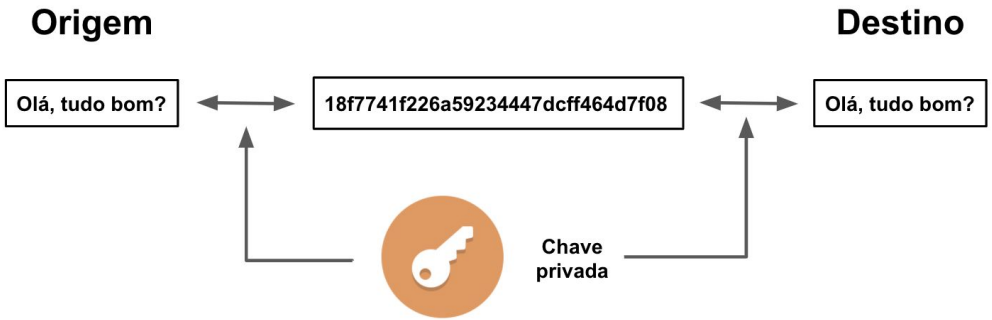
O que é criptografia?

- A criptografia pode ser usada para proteger dados em repouso, em trânsito ou durante o processamento.

- **Ela tem vários objetivos, como:**
  - Confidencialidade: garantir que as informações sejam disponibilizadas apenas para usuários autorizados
  - Integridade: assegurar que as informações não tenham sido alteradas
  - Autenticação: confirmar a autenticidade das informações ou a identidade do usuário
  - Não repúdio: impedir que o usuário negue compromissos ou ações anteriores

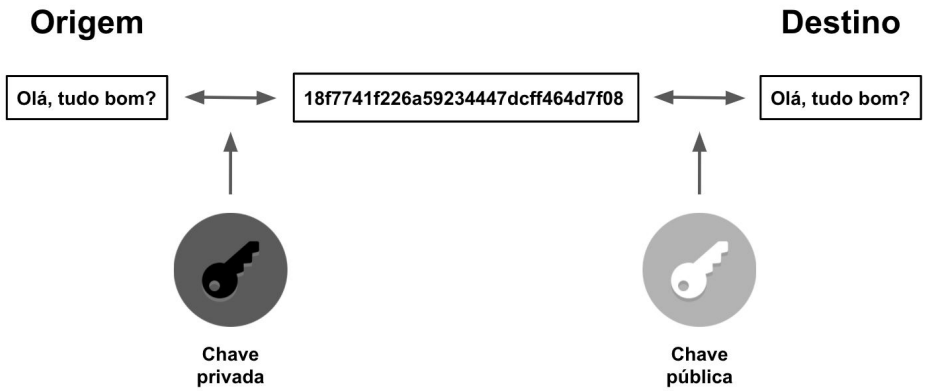
Tipos de criptografia

- Criptografia simétrica: Utiliza a mesma chave para criptografar e descriptografar.



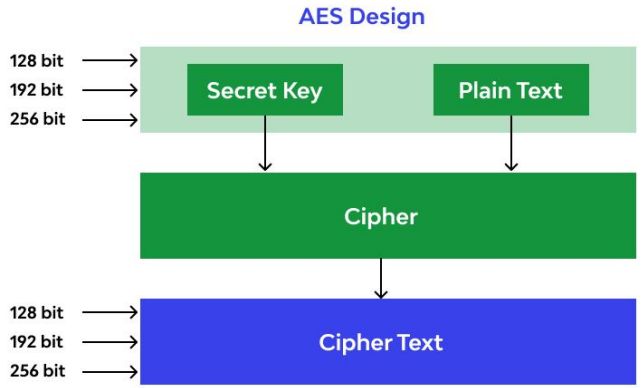
Tipos de criptografia

- Criptografia assimétrica: Utiliza duas chaves separadas, uma pública e uma privada. A chave pública é compartilhada entre todas as partes, mas apenas quem tiver a chave privada poderá descriptografar.



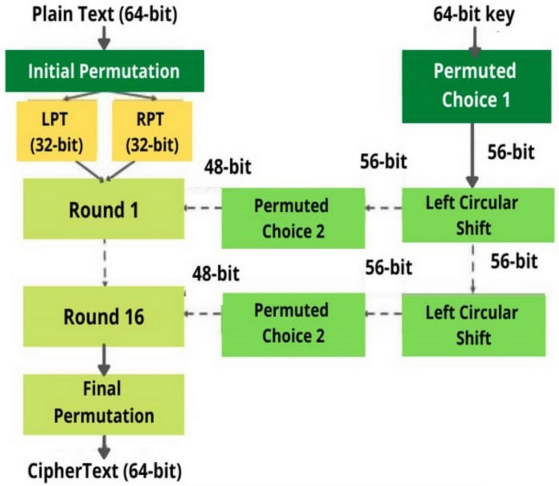
Encriptação Simétrica - Principais algoritmos

- **AES (Advanced Encryption Standard).**
  - Usa uma única chave para criptografar e descriptografar os dados. O comprimento da chave pode ser de 128, 192 ou 256 bits, influenciando o nível de segurança.
  - Opera em blocos de 128 bits (16 bytes), independentemente do tamanho do dado. Dados maiores são divididos em blocos, e os menores são preenchidos (padding).
  - AES é amplamente adotado em protocolos de segurança como HTTPS, VPNs e redes sem fio seguras, devido à sua eficiência e robustez.



Encriptação Simétrica - Principais algoritmos

- **DES (menos usado, inseguro atualmente).**
  - Usa uma única chave de 56 bits para criptografar e descriptografar os dados. Essa chave é curta para os padrões atuais, o que a torna vulnerável a ataques de força bruta.
  - Opera em blocos de 64 bits, ou seja, os dados são divididos em partes de 64 bits e processados individualmente.
  - DES foi amplamente substituído pelo AES, mas ainda pode ser encontrado em sistemas legados. É considerado inseguro para uso em sistemas modernos.



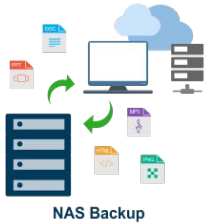
Encriptação e Decriptação Simétrica em python

- Usando a biblioteca **cryptography**.
- **Gerar Chave:** Gera uma chave única para criptografia e inicializa um objeto de cifra (Fernet).
- **Ler Arquivo Original:** Lê o conteúdo do arquivo arquivo.txt em formato binário.
- **Criptografar Dados:** Criptografa o conteúdo do arquivo usando a chave gerada.
- **Salvar Dados Criptografados:** Escreve os dados criptografados em um novo arquivo chamado arquivo_encrypted.txt.
- **Ler Dados Criptografados:** Lê o conteúdo do arquivo criptografado.
- **Descriptografar Dados:** Descriptografa os dados utilizando a mesma chave.
- **Salvar Dados Descriptografados:** Escreve os dados descriptografados em um novo arquivo chamado arquivo_decrypted.txt.

```python
from cryptography.fernet import Fernet

# Gerar chave
key = Fernet.generate_key()
cipher_suite = Fernet(key)

# Encriptar arquivo
with open('arquivo.txt', 'rb') as file:
    data = file.read()
encrypted_data = cipher_suite.encrypt(data)

# Salvar arquivo encriptado
with open('arquivo_encrypted.txt', 'wb') as file:
    file.write(encrypted_data)

# Decriptar arquivo
with open('arquivo_encrypted.txt', 'rb') as file:
    encrypted_data = file.read()
decrypted_data = cipher_suite.decrypt(encrypted_data)

# Salvar arquivo decriptado
with open('arquivo_decrypted.txt', 'wb') as file:
    file.write(decrypted_data)
```

Aplicações práticas

- **Proteção de Dados em Trânsito**
  - Exemplo: Comunicação em redes Wi-Fi.
  - Aplicação: Encriptação de dados enviados entre dispositivos para evitar interceptações, como em conexões HTTPS (SSL/TLS).



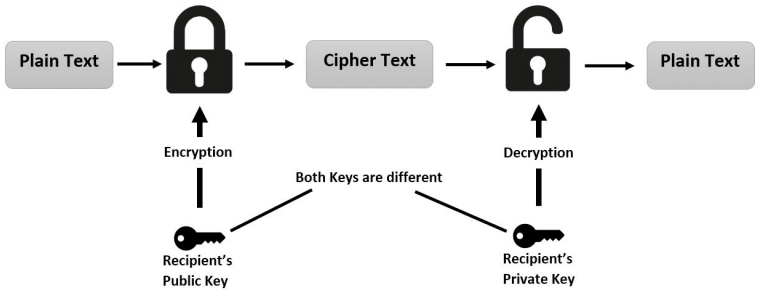
- **Armazenamento Seguro de Dados**
  - Exemplo: Proteção de arquivos confidenciais.
  - Aplicação: Arquivos armazenados localmente ou em nuvens são encriptados para garantir que apenas pessoas autorizadas possam acessá-los.

- **Backups de Dados**
  - Exemplo: Backups de sistemas corporativos.
  - Aplicação: Backups encriptados garantem que informações críticas não sejam acessadas por invasores, mesmo que o armazenamento seja comprometido.

Encriptação Assimétrica - Principais algoritmos

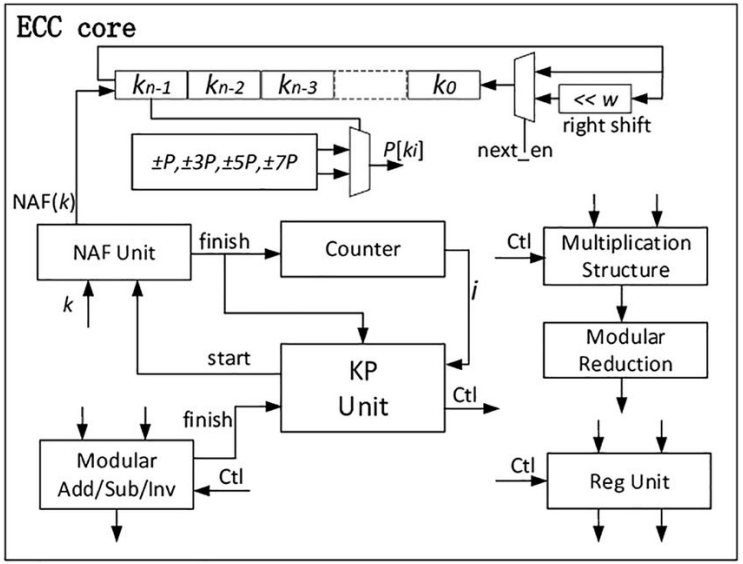
- **RSA (Rivest-Shamir-Adleman).**
  - Uma chave pública para encriptar.
  - Uma chave privada para descriptografar.

- **Funcionamento Simplificado:**
  - Gera duas chaves com base em números primos grandes.
  - Encripta dados com a chave pública.
  - Apenas quem tem a chave privada pode descriptografar.



Encriptação Assimétrica - Principais algoritmos

- **ECC (Elliptic Curve Cryptography).**
  - É um método de criptografia assimétrica que usa propriedades de curvas elípticas para gerar chaves.
  - Ele oferece:
    - Alta segurança com chaves menores (ex.: 256 bits em ECC ≈ 3072 bits em RSA).
    - **Eficiência:** Menor uso de processamento, ideal para dispositivos com recursos limitados.
    - **Aplicações:** HTTPS, assinaturas digitais (ECDSA), troca de chaves (ECDH) e criptografia em IoT e dispositivos móveis.



Encriptação e Decriptação Assimétrica em python

```python
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization

# Gerar par de chaves
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
public_key = private_key.public_key()

# Salvar chave privada
with open("private_key.pem", "wb") as file:
    file.write(private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    ))

# Salvar chave pública
with open("public_key.pem", "wb") as file:
    file.write(public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    ))
```

```python
# Encriptar arquivo
with open('arquivo.txt', 'rb') as file:
    data = file.read()
encrypted_data = public_key.encrypt(
    data,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

# Salvar arquivo encriptado
with open('arquivo_encrypted.txt', 'wb') as file:
    file.write(encrypted_data)

# Decriptar arquivo
with open('arquivo_encrypted.txt', 'rb') as file:
    encrypted_data = file.read()
decrypted_data = private_key.decrypt(
    encrypted_data,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

# Salvar arquivo decriptado
with open('arquivo_decrypted.txt', 'wb') as file:
    file.write(decrypted_data)
```

Encriptação e Decriptação Assimétrica em python

- **Gerar Chaves RSA:**
  - Cria uma chave privada e sua correspondente chave pública.

- **Salvar Chaves:**
  - Salva a chave privada no arquivo private_key.pem e a chave pública no arquivo public_key.pem em formato PEM.

- **Encriptar Arquivo:**
  - Lê o conteúdo de arquivo.txt e o encripta usando a chave pública com o esquema de padding OAEP e o hash SHA-256.
  - Salva os dados encriptados no arquivo arquivo_encrypted.txt.

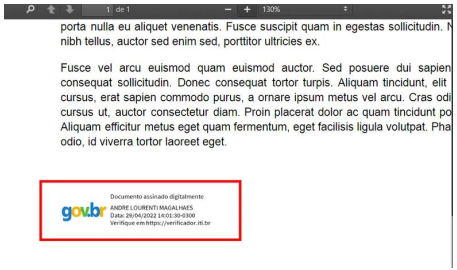
- **Decriptar Arquivo:**
  - Lê o conteúdo de arquivo_encrypted.txt e o decripta usando a chave privada.
  - Salva os dados decriptados no arquivo arquivo_decrypted.txt.

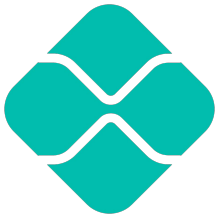
Aplicações práticas

- **Certificados Digitais**
  - Exemplo: HTTPS, SSL/TLS.
  - Aplicação: Garante a autenticidade de servidores e navegadores, protegendo a comunicação com sites.

- **Assinaturas Digitais**
  - Exemplo: Documentos oficiais, contratos eletrônicos.
  - Aplicação: Valida a autenticidade e integridade de um documento, garantindo que ele foi assinado pela parte correta.

- **Sistemas de Pagamento**
  - Exemplo: Cartões de crédito e transações bancárias.
  - Aplicação: Garante que os dados financeiros sejam protegidos durante as transações online.





Comparação: Simétrica vs Assimétrica

Aspecto	Simétrica	Assimétrica
Chaves	Única	Par de chaves (pública e privada)
Velocidade	Mais rápida	Mais lenta
Uso típico	Encriptação de dados em massa	Comunicação segura e autenticação

Referências

- https://pypi.org/project/cryptography/
- TKOTZ, Viktoria. Criptografia–Segredos Embalados para Viagem. Novatec Editora, 2024.
- WRIGHTSON, Tyler. Segurança de redes sem fio: guia do iniciante. Bookman Editora, 2014.

Obrigado!

Dúvidas?

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br