# Criptografia e Funções Hash

## Introdução

Neste trabalho vamos falar de alguns tipos de segurança usados em computadores. O objetivo é entender e mostrar exemplos de como funcionam a criptografia e as funções hash, usando a linguagem Python. Vamos ver três pontos principais:  
- Criptografia com Chaves Simétricas;  
- Criptografia com Chaves Assimétricas;  
- Função Hash.

## Criptografia com Chaves Simétricas

Na criptografia simétrica, usamos a mesma chave (uma espécie de senha) para criptografar e também para descriptografar a mensagem. É como se fosse uma chave de casa que abre e fecha a mesma porta. Esse tipo de criptografia é rápido, mas precisa que a chave seja guardada em segredo.

Exemplo em Python:

from cryptography.fernet import Fernet  
  
# Geração da chave  
chave = Fernet.generate\_key()  
fernet = Fernet(chave)  
  
# Mensagem original  
mensagem = "Segurança de Dados".encode()  
  
# Criptografia  
mensagem\_criptografada = fernet.encrypt(mensagem)  
print("Criptografada:", mensagem\_criptografada)  
  
# Descriptografia  
mensagem\_descriptografada = fernet.decrypt(mensagem\_criptografada)  
print("Descriptografada:", mensagem\_descriptografada.decode())

## Criptografia com Chaves Assimétricas

Na criptografia assimétrica temos duas chaves diferentes:  
- A chave pública serve para criptografar (trancar a mensagem).  
- A chave privada serve para descriptografar (abrir a mensagem).  
  
Esse modelo é mais seguro, porque mesmo que alguém veja a chave pública, não consegue abrir a mensagem sem a chave privada.

Exemplo em Python:

from cryptography.hazmat.primitives.asymmetric import rsa, padding  
from cryptography.hazmat.primitives import hashes  
  
# Geração do par de chaves  
chave\_privada = rsa.generate\_private\_key(public\_exponent=65537, key\_size=2048)  
chave\_publica = chave\_privada.public\_key()  
  
mensagem = b"Mensagem secreta RSA"  
  
# Criptografia com a chave pública  
mensagem\_criptografada = chave\_publica.encrypt(  
 mensagem,  
 padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),  
 algorithm=hashes.SHA256(),  
 label=None)  
)  
  
print("Criptografada:", mensagem\_criptografada)  
  
# Descriptografia com a chave privada  
mensagem\_descriptografada = chave\_privada.decrypt(  
 mensagem\_criptografada,  
 padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),  
 algorithm=hashes.SHA256(),  
 label=None)  
)  
  
print("Descriptografada:", mensagem\_descriptografada.decode())

## Função Hash

As funções hash são diferentes da criptografia. Elas não servem para esconder e depois mostrar de novo. Uma vez que transformamos uma informação em hash, não dá para voltar ao que era antes. Por isso, elas são usadas para guardar senhas de forma segura e também para verificar se arquivos foram alterados.

Exemplo em Python (SHA-256):

import hashlib  
  
senha = "minhaSenhaSegura123"  
hash\_senha = hashlib.sha256(senha.encode()).hexdigest()  
  
print("Senha original:", senha)  
print("Hash SHA-256:", hash\_senha)

## Conclusão

Neste trabalho vimos três formas de segurança:  
- Criptografia simétrica: rápida, mas depende de guardar bem a chave.  
- Criptografia assimétrica: mais segura, porque usa duas chaves diferentes.  
- Funções hash: não podem ser revertidas e servem muito para senhas.  
  
Essas técnicas são muito importantes para a proteção de dados na internet e no dia a dia.

João Pedr Martins Carvalho

RA 825115359