Exerc´ıcios Simples sobre Criptografia HASH e Fernet com Python e MongoDB

Bruno Pera

September 24, 2024

Lista de Exerciıcios

1. Gerar hash de senhas com SHA-256:

Escreva um programa que receba uma senha do usuário e gere o hash usando o algoritmo SHA-256, imprimindo o resultado.

import hashlib

senha = input("Digite sua senha:")

hash\_obj = hashlib.sha256(senha.encode())

print(hash\_obj)

2. Verificar se duas strings tem o mesmo hash:

Escreva um programa que receba duas strings e gere os hashes de ambas. Verifique se os hashes gerados são iguais ou diferentes.

import hashlib

str = input("Digite alguma coisa na string 1:")

str2 = input("Digite alguma coisa na string 2:")

hash\_obj = hashlib.sha256(str.encode()).hexdigest()

hash\_obj2 = hashlib.sha256(str2.encode()).hexdigest()

if hash\_obj == hash\_obj2 :

print(True)

else:

print(False)

3. Salvar hash de uma string no MongoDB:

Crie um programa que receba uma string, gere o hash com SHA-256 e salve o hash no MongoDB.

import hashlib

from pymongo.mongo\_client import MongoClient

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

str = input("Digite alguma coisa na string :")

hash\_obj = hashlib.sha256(str.encode()).hexdigest()

colecao.insert\_one({"hash" : hash\_obj})

4. Comparar hash de uma string com valor armazenado no Mon goDB:

Escreva um programa que receba uma string, gere seu hash e compare-o com um hash previamente armazenado no MongoDB.

import hashlib

from pymongo.mongo\_client import MongoClient

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

str = input("Digite alguma coisa na string :")

hash\_obj = hashlib.sha256(str.encode()).hexdigest()

hash\_colecao = colecao.find\_one({"hash" : hash\_obj})

if hash\_colecao :

print(True)

else:

print(False)

5. Gerar hash com MD5:

Modifique o programa do exercício 1 para gerar o hash da senha utilizando o algoritmo MD5 em vez de SHA-256

import hashlib

senha = input("Digite sua senha: ")

hash\_obj = hashlib.md5(senha.encode()).hexdigest()

print("Hash MD5 da senha:", hash\_obj)

.

6. Comparar hashes gerados por diferentes algoritmos: Crie um programa que gere os hashes de uma mesma string utilizando MD5, SHA-1 e SHA-256. Compare e mostre as diferenças entre os valores gerados.

import hashlib

senha = input("Digite sua senha: ")

hash256\_obj = hashlib.sha256(senha.encode()).hexdigest()

hash1\_obj = hashlib.sha1(senha.encode()).hexdigest()

md5\_obj = hashlib.md5(senha.encode()).hexdigest()

print("Hash sha256 da senha:", hash256\_obj)

print("Hash MD5 da senha:", md5\_obj)

print("Hash sha1 da senha:", hash1\_obj)

7. Criptografar uma mensagem com Fernet:

Utilizando a biblioteca cryptography, crie uma chave Fernet e use-a para criptografar uma mensagem fornecida pelo usuário. Imprima a mensagem criptografada.

import hashlib

from cryptography.fernet import Fernet

chave = Fernet.generate\_key()

Chavefernet = Fernet(chave)

valor = input("Digite um valor: ")

val\_criptografado = Chavefernet.encrypt(valor.encode())

print(val\_criptografado)

8. Descriptografar uma mensagem com Fernet:

Escreva um programa que descriptografe uma mensagem criptografada usando Fernet, exibindo a mensagem original.

import hashlib

from cryptography.fernet import Fernet

chave = Fernet.generate\_key()

Chavefernet = Fernet(chave)

valor = input("Digite um valor: ")

val\_criptografado = Chavefernet.encrypt(valor.encode())

print(f"Criptografado >>> {val\_criptografado}")

val\_descript = Chavefernet.decrypt(val\_criptografado).decode()

print(f"Descriptografado >>> {val\_descript}")

1

9. Armazenar uma mensagem criptografada com Fernet no MongoDB:

Crie um programa que criptografe uma mensagem com Fernet e armazene a mensagem criptografada no MongoDB.

import hashlib

from cryptography.fernet import Fernet

from pymongo.mongo\_client import MongoClient

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

chave = Fernet.generate\_key()

Chavefernet = Fernet(chave)

valor = input("Digite um valor: ")

val\_criptografado = Chavefernet.encrypt(valor.encode())

colecao.insert\_one({"Criptografia fernet" : val\_criptografado})

10. Recuperar e descriptografar uma mensagem do MongoDB: Escreva um programa que recupere uma mensagem criptografada armazenada no MongoDB e a descriptografe usando a chave Fernet.

import hashlib

from cryptography.fernet import Fernet

from pymongo.mongo\_client import MongoClient

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

chave = Fernet.generate\_key()

Chavefernet = Fernet(chave)

id = input("Digite um id: ")

busca= colecao.find\_one({"\_id": id})

resultado\_criptografado = busca["Criptografia fernet"]

resultado\_descriptografado = Chavefernet.decrypt(resultado\_criptografado).decode()

print(resultado\_descriptografado)

11. Gerar e salvar uma chave Fernet:

Crie um programa que gere uma chave Fernet e a armazene em um arquivo local para uso futuro.

from cryptography.fernet import Fernet

chave = Fernet.generate\_key()

arquivo = open('C:/Users/PC/Desktop/python-projects/Listas/chaveFernet.key', 'wb')

arquivo.write(chave)

print("Arquivo criado com sucesso!")

12. Ler uma chave Fernet de um arquivo e criptografar uma men sagem:

Modifique o programa do exercício 7 para que ele leia a chave Fernet de um arquivo ao invés de gerar uma nova chave toda vez.

import hashlib

from cryptography.fernet import Fernet

arquivo = open('C:/Users/PC/Desktop/python-projects/Listas/chaveFernet.key', 'rb')

chavetxt = arquivo.readline()

Chavefernet = Fernet(chavetxt)

valor = input("Digite um valor: ")

val\_criptografado = Chavefernet.encrypt(valor.encode())

print(val\_criptografado)

13. Armazenar a chave Fernet no MongoDB:

Crie um programa que gere uma chave Fernet e armazene a chave no MongoDB para uso posterior.

import hashlib

from cryptography.fernet import Fernet

from pymongo.mongo\_client import MongoClient

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

chave = Fernet.generate\_key()

colecao.insert\_one({"Chave fernett":chave})

14. Recuperar uma chave Fernet do MongoDB e descriptografar uma mensagem:

Escreva um programa que recupere uma chave Fernet armazenada no Mon goDB e a utilize para descriptografar uma mensagem.

import hashlib

from cryptography.fernet import Fernet

from pymongo.mongo\_client import MongoClient

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

from bson import ObjectId # Importa o ObjectId

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

chave = colecao.find\_one({"\_id": ObjectId("67085e0c3d674f33139852b8")})

chave = chave["Chave fernett"]

chave = Fernet(chave)

valor = input("Digite qualquer coisa: ")

valor\_crpitografado = chave.encrypt(valor.encode())

valor\_descriptografado = chave.decrypt(valor\_crpitografado).decode()

print(f"A mensagem descriptografada = {valor\_descriptografado}")

gAAAAABnCn5FdDAt084aAl5GrPhxGysbxbhRjZLFb5bDlnXDgvzC4r3erJ49aeeXj\_pcdivM32vVTIK5LyJAIEFrHqlsY\_hGcWA3K1pT4D2-bp357uVIR-4=

15. Criptografar e descriptografar arquivos com Fernet: Escreva um programa que criptografe um arquivo de texto utilizando Fer net e, em seguida, implemente outro programa para descriptografar esse arquivo.

criptografia :

from cryptography.fernet import Fernet

def leitura\_chave(metodo = ""):

chavetxt = open('C:/Users/PC/Desktop/python-projects/Listas/chaveFernet.key', metodo)

return chavetxt.read()

def leitura\_arquivo(metodo = ""):

chavetxt = open('C:/Users/PC/Desktop/python-projects/Listas/mmensagem.txt', metodo)

return chavetxt.read()

def criptografar (texto):

arquivobites = open('C:/Users/PC/Desktop/python-projects/Listas/mensagem.txt', 'wb')

conteudo\_Criptografado = chave.encrypt(texto.encode())

arquivobites.write(conteudo\_Criptografado)

arquivobites.close()

chave = leitura\_chave ("rb")

chave = Fernet(chave)

arquivotxt = leitura\_arquivo("r+")

criptografar(arquivotxt)

descriptografia :

from cryptography.fernet import Fernet

def leitura\_chave(metodo = ""):

chavetxt = open('C:/Users/PC/Desktop/python-projects/Listas/chaveFernet.key', metodo)

return chavetxt.read()

def leitura\_arquivo(metodo = ""):

txtbytes = open('C:/Users/PC/Desktop/python-projects/Listas/mensagem.txt', metodo)

print ()

return txtbytes.read()

def descriptografia (textobytes):

arquivobites = open('C:/Users/PC/Desktop/python-projects/Listas/mensagem.txt', 'w')

conteudo\_descriptografado = chave.decrypt(textobytes).decode()

arquivobites.write(conteudo\_descriptografado)

arquivobites.close()

chave = leitura\_chave ("rb")

chave = Fernet(chave)

arquivo\_textobytes = leitura\_arquivo("rb")

descriptografia(arquivo\_textobytes)

16. Comparar o tamanho de diferentes hashes:

Escreva um programa que gere hashes de diferentes strings com MD5, SHA-1 e SHA-256. Compare o tamanho dos hashes gerados por cada algoritmo.

import hashlib

senha = input("Digite sua senha: ")

hash256\_obj = hashlib.sha256(senha.encode()).hexdigest()

hash1\_obj = hashlib.sha1(senha.encode()).hexdigest()

md5\_obj = hashlib.md5(senha.encode()).hexdigest()

print("Hash sha256 da senha:", len(hash256\_obj) , "caractes criptografados" )

print("Hash MD5 da senha:", len(md5\_obj) , "caractes criptografados")

print("Hash sha1 da senha:", len(hash1\_obj) , "caractes criptografados")

17. Exibir o hash hexadecimal de uma string:

Modifique o programa do exerc´ıcio 1 para exibir o hash gerado em formato hexadecimal.

import hashlib

senha = input("Digite sua senha:")

hash\_obj = hashlib.sha256(senha.encode()).hexdigest()

print(hash\_obj)

18. Validar uma senha usando hash:

Crie um programa que receba uma senha, gere seu hash e armazene-o no MongoDB. Depois, receba uma senha do usuário e compare seu hash com o valor armazenado

import hashlib

from pymongo.mongo\_client import MongoClient

senha = input("Digite sua senha:")

hash\_obj = hashlib.sha256(senha.encode())

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

colecao.insert\_one({"hash" : hash\_obj.hexdigest() })

senha = input("Digite sua senha novamente:")

hash\_obj2 = hashlib.sha256(senha.encode())

resultado = colecao.find\_one({"hash": hash\_obj2.hexdigest()})

if resultado:

print("Autenticação bem-sucedida!")

else:

print("Senha incorreta.")

.

19. Criptografar múltiplas mensagens com Fernet:

Escreva um programa que criptografe várias mensagens fornecidas pelo usuário e armazene-as no MongoDB.

import hashlib

from cryptography.fernet import Fernet

from pymongo.mongo\_client import MongoClient

uri = "mongodb+srv://root:123@cluster0.bqxvu.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"

client = MongoClient(uri)

db = client['projetobruno']

colecao = db['aluno']

chave = Fernet.generate\_key()

Chavefernet = Fernet(chave)

resp = ''

resp = input("Digite um valor: | espaco para parar ")

texto = []

while resp !="":

texto.append(resp)

resp = input("Digite um valor: | espaco para parar ")

texto\_completo = ' '.join(texto)

val\_criptografado = Chavefernet.encrypt(texto\_completo.encode())

colecao.insert\_one({"Criptografia fernet" : val\_criptografado})

20. Verificar integridade de dados com hash:

Crie um programa que receba um arquivo, gere seu hash com SHA-256, e depois de uma alteração no arquivo, gere o hash novamente para verificar se o arquivo foi alterado.

import hashlib

import os

def gerar\_hash\_sha256(caminho\_arquivo):

arquivo = open('C:/Users/PC/Desktop/python-projects/Listas/mensagem.txt , "rb")

sha256\_hash = hashlib.sha256()

for byte\_block in iter(lambda: arquivo.read(4096), b""):

sha256\_hash.update(byte\_block)

return sha256\_hash.hexdigest()

caminho = input("Digite o caminho do arquivo: ")

hash\_inicial = gerar\_hash\_sha256(caminho)

hash\_atual = gerar\_hash\_sha256(caminho)

if hash\_inicial == hash\_atual:

print("O arquivo NÃO foi alterado.")

else:

print("O arquivo FOI alterado!")

3