Exercices

Question 1:

Vous devez implémenter un programme Python combinant le hachage SHA-256 et HMAC pour une authentification sécurisée des données. Votre défi consiste à créer une classe qui prend un message saisi par l'utilisateur, calcule le hachage SHA-256 du message, puis génère un HMAC à l'aide du hachage SHA-256 et d'une clé secrète. Votre objectif est de garantir l'intégrité et l'authenticité du message.

Votre code devrait suivre programmation orientée objet (POO) avec les classes et les objets !

Résultat :

Enter a message to authenticate: How are you today?
Original Message: How are you today?
SHA-256 Hash: 68514b65884697a4725d4b231dd376aa745abfd4feb3cf9aaa21e4194e2b8acc
HMAC: 82e43e62fcbebd8ad81857fcaa766516b6b5caa300912b4d4e3afebf4fb227af

Question 2 :

Vous êtes chargé de mettre en œuvre un système de cryptage et de déchiffrement de base des messages à l'aide d'AES en Python. Votre objectif est de créer un programme simple qui prend un message **saisi** par **l'utilisateur**, le crypter à l'aide d'AES, puis le déchiffrer pour lui redonner sa forme originale. Utilisez le modèle de démonstration de classe AES.

Votre code devrait suivre la programmation orientée objet (POO) avec les classes et les objets !

Résultat :

Enter a message to encrypt: Hello, Welcome to the Cybersecurity!!!

Encrypted Message: b'\x8c\x17F\xf9\xbcO\xcbL\x90)\xea\x8f

{\xfd\xb1\x8b\x06D\xf6\xf3\x11\\\xb0\xd6w\x9f\x8b#k\xf9b\xd5\xda_\xe9\xe9\xfa\xfaV\x05\xa1\x0b\xe1H\t\xf4^'

Decrypted Message: Hello, Welcome to the Cybersecurity!!!

Question 3 :

Vous devez implémenter un système de cryptage et de déchiffrement RSA de base en Python. Votre tâche consiste à créer un programme Python qui prend un message saisi par l'utilisateur, génère une paire de clés RSA, crypter le message à l'aide du schéma de remplissage RSA, PKCS1_OAEP, puis le déchiffrer dans sa forme d'origine.

Votre code devrait suivre la programmation orientée objet (POO) avec les classes et les objets !

Résultat :

Enter a message to encrypt: Hello, you are getting good with python!

Encrypted Message:
b"\x8c\x1a\xd1Q@\xce\xd0a8\xb4\x1d\xc4\xa0\x80n\xfd|\xbaZt!,\xe7\x9e\xff\xd0\xe7?\x11r\xf8D
\xe8\xc1\xb9\x8f\xb6t\xaa/\xcb\x14\xb8\xd6\x94,\x15y\\\xf6'\xba\xfad\xb7\x91\xa9\xa5\xb1<\xa0
8\x0f\x0e\x1d\xcf\xf4\xe5\x15k\xa8\x95\x87\xdeZ\x13\x1c|^B\x97c\xd3\xba\x073\xef\xee\xab\xb
4\xae\xad\xb4\x89@[\x0b\xbdl\x88\xaa\xf3-
\x1d\x98C\r}\xda\xaf\xc6\xc7\xf3xN\xd7\x89[N\x9a\xba\xe2\x05\x19*\x86\xa3W\r\xb6\xf6S\xa2J2
\xbcw\n\xa1\xcaJ\xc2\xed\xc0\x00\xc6\x15\x86\xa7\xce\x11U\x19\n\x02~\x8b*^\xc45\xfe\x1e;~\x
83\x98\xe6wJ*\xb5Ke\x90\xe0[\xb7\xc3e\x1d
6>\x13\xe6\xa5'7\xdd\x93\x8e\x1cd\x1f\xb3\xbc\xf5z\x9c\x10\xe0\xdf\x19\xf4H}\xe1-
\xfb\xb4M\x0em\x9a\x80\xeb\x8b\x9f\xb1\xceu\x02u>U\xa9\xc9(\xac\x84h{ks\n\x18\xe7\xaf\xa6J
K\xa5\r?|\x9d\x07\x9fE\xd2\x1e\xa6\xcc\x8a\x8e"

Decrypted Message: Hello, you are getting good with python!

Question 4 :

Votre défi consiste à créer un programme Python qui permet aux utilisateurs de saisir des messages à plusieurs reprises, de chiffrer et déchiffrer ces messages et de stocker les données chiffrées dans un fichier JSON. Le programme doit continuer à s'exécuter jusqu'à ce que l'utilisateur le ferme (en utilisant Ctrl + C).

Astuce : exemples de packages, n'hésitez pas à en utiliser :

```
import json

from Cryptodome.Cipher import AES

from Cryptodome.Random import get_random_bytes

from Cryptodome.Util.Padding import pad, unpad
```

Votre code devrait suivre la programmation orientée objet (POO) avec les classes et les objets !

Résultat :

```
Enter a message to encrypt (Ctrl + C to exit): hello
Encrypted Message: 76cc52701756772ef4f2fb5dcd7f9fce
Decrypted Message: hello
Enter a message to encrypt (Ctrl + C to exit): how are you?
Encrypted Message: dfc1af7bb741e1f1a93f60e24d742420
Decrypted Message: how are you?
Enter a message to encrypt (Ctrl + C to exit): Welcome
Encrypted Message: a1e4fa710d4baf066c25da8a5f8d7476
Decrypted Message: Welcome
Enter a message to encrypt (Ctrl + C to exit):
Program terminated.
```

In json file output:


```
{"original_message": "hello", "encrypted_message": "76cc52701756772ef4f2fb5dcd7f9fce", "iv": "7f5a4334288a6f3d883e0cc3748dce5d"}
{"original_message": "how are you?", "encrypted_message": "dfc1af7bb741e1f1a93f60e24d742420", "iv": "e0db33f42cd1ed0c6dd97caafb4ed42d"}
{"original_message": "Welcome", "encrypted_message": "a1e4fa710d4baf066c25da8a5f8d7476", "iv": "6d67540630cfbaa1d6ced0641d880dc5"}
```


Question 5 :

Écrivez un programme Python qui gère les messages en toute sécurité en calculant le hachage SHA-256, en générant un HMAC, en chiffrant le message à l'aide d'AES et en stockant les données dans une base de données SQLite.

Le programme doit inviter l'utilisateur à saisir un message, puis calculer le hachage SHA-256, générer un HMAC, chiffrer le message à l'aide d'une clé secrète générée aléatoirement pour AES. Enfin stocker le message d'origine, le hachage SHA-256, HMAC, message chiffré et vecteur d'initialisation (IV) dans une base de données SQLite nommée secure_data.db.

Le programme doit fournir des commentaires à l'utilisateur en imprimant le message original, le hachage SHA-256, le HMAC, le message chiffré et le message déchiffré. De plus, gérez les interruptions des utilisateurs (par exemple, KeyboardInterrupt) et assurez-vous que la connexion SQLite est fermée à la fin du programme.

Votre code devrait suivre la programmation orientée objet (POO) avec les classes et les objets !

Résultat :

Enter a message to secure (Ctrl + C to exit): Hello Good Afternoon!
Original Message: Hello Good Afternoon!
SHA-256 Hash: 6f721605c42e4666c2f279574fa7e235d05b2f86b00fd771434f4d0c94fe20a8
HMAC: ff7f23e30b1029bb433b62c532905c7a3391b05eb7762e93003f3dbfb3f598da
Encrypted Message: 4e558bc2e67fcc4fa86d8884eea1317acf92404faba4f36dc1a4f0faed75c0bb
Decrypted Message: Hello Good Afternoon!

Enter a message to secure (Ctrl + C to exit): Hi I'm good
Original Message: Hi I'm good
SHA-256 Hash: 66d22e5cdc6edfa26b36f1aa17782d9a892bafd54c4027a425d93213f0ce821f
HMAC: e2d80dcdf0f3e55fde2c7ee3e199fa130fabfe42499bbaa9a54e13941dbcbb73
Encrypted Message: fe37262c6e82982c1350d4131bcf6d8e
Decrypted Message: Hi I'm good
Enter a message to secure (Ctrl + C to exit):
Program terminated.

	id	original_message	sha256_hash	hmac	encrypted_message	iv
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Hello Good Afternoon!	6f721605c42e4666c2f279574fa7e235d05b2f86b0...	ff7f23e30b1029bb433b62c532905c7a3391b05eb...	4e558bc2e67fcc4fa86d8884eea1317acf92404fab...	b5d86169d8cd93a8dafdaebd057abceb
2	2	Hi I'm good	66d22e5cdc6edfa26b36f1aa17782d9a892bafd54...	e2d80dcdf0f3e55fde2c7ee3e199fa130fabfe42499...	fe37262c6e82982c1350d4131bcf6d8e	56f4c42e36aeccc4501ac6f2168c82a2