Final Project: Part 2 - Secure Information Using Symmetric Encryption

In Part 2 of this final project, **Secure Information Using Symmetric Encryption**, you are required to complete the steps in the provided Cloud IDE workspace and upload the corresponding screenshot for each step to the designated area in the platform. You should plan to complete Part 2 in about 30 minutes.

Peer-Graded Final Project: Part 2 - Secure Information Using Symmetric Encryption

Estimated time needed: 30 minutes

In **Part 2** of the final project, you will demonstrate your ability to protect **SecureBank's** information by implementing appropriate symmetric encryption techniques. As part of this project, you will also review a peer's project submission using the rubric provided.

Part 2 consists of one task:

- **Task 4:** Securing information using symmetric encryption

Step-by-step instructions

Refresh your knowledge

Before starting on Part 2 of this project, you can refresh your knowledge by referring to the following lesson:

- **Module 5, Lesson 1:** Introduction to Cryptography

Project scenario

Imagine you have recently been hired as an ethical hacker by SecureBank, a mid-sized financial institution known for its rapidly growing digital services. In the wake of recent incidents in the banking sector, the management has become increasingly anxious about potential vulnerabilities in SecureBank's online banking platform. Preliminary investigations have highlighted weak security measures, including inadequate cryptographic protections for sensitive financial data. To address these risks, the bank has

tasked you with strengthening its security posture through advanced cryptographic techniques.

Objectives

In this assignment, you will implement symmetric encryption to secure sensitive information at SecureBank by:

- Encrypting financial data using symmetric encryption techniques (e.g., AES)
- Decrypting encrypted data to ensure it can be accessed securely
- Managing encryption keys properly to protect against unauthorized access
- Demonstrating the implementation of encryption and decryption in a secure environment

Note about screenshots

Throughout this lab, you will be prompted to take screenshots and save them on your device. You will need to upload the screenshots for peer review. You can use various free screen grabbing tools or your operating system's shortcut keys (Alt + PrintScreen in Windows, for example) to capture the required screenshots. You can save the screenshots with the .jpg or .png extension.
Confirm that the screenshot clearly displays the result of your activity. It's recommended that you set your system display resolution to the highest available setting.

Task 4/4: Secure information using symmetric encryption

***Note:** Ensure to complete Part 2 of the final project, Secure Information Using Symmetric Encryption.

Step 1: Create the "userdetails" file

Sample credentials: You are provided with the following credentials that need to be stored in the "userdetails" file:

Username: admin

Password: P@ssw0rd123

Email: admin@example.com

Phone: 123-456-7890

Prompt: Create the "userdetails" file using any Linux-based tools/commands. Then, display the content of the file using the "cat" command.

Ensure the screenshot clearly shows the key detail outlined below.

Key detail:

- The contents of the "userdetails" file

Create and display contents of user details file

The userdetails file was created using the echo command to input sample credentials (username, password, email, and phone). The contents were then displayed using the cat command to verify successful creation in the terminal. All required fields are present and formatted as specified.

Task 4/4: Secure information using symmetric encryption

Step 2: Generate the Advanced Encryption Standard (AES) random key

Prompt: Create a random encryption key for AES encryption and store it in the "passkey" file. Ensure the screenshot clearly shows the key details outlined below.

Key details:

- Successful execution of the key generation command
- A list of files confirming the creation of the "passkey" file

AES Key Generation and Passkey File Confirmation

The screenshot demonstrates the successful creation of random AES encryption key using the command openssl rand -base64 32 > passkey. The output of ls -l confirms the existence and correct permissions of both the passkey and userdetails files in the terminal environment, satisfying the project's key file creation requirements.

Task 4/4: Secure information using symmetric encryption

Step 3: Encrypt the “userdetails” file using passkey

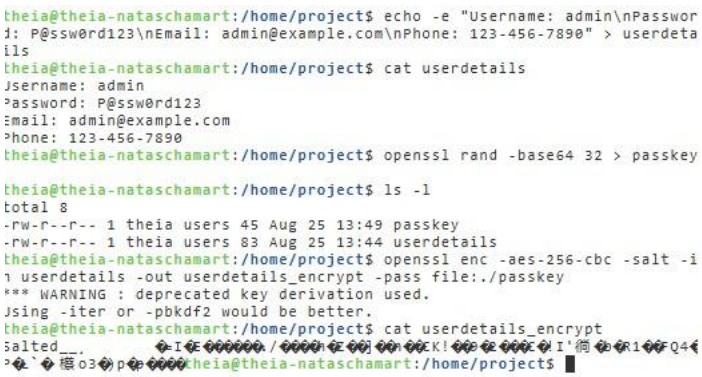
Prompt: Use the generated key to encrypt the contents of the “userdetails” file and create an encrypted file named “userdetails_encrypt.” Ensure the screenshot clearly shows the key details outlined below.

Key details:

- Successful execution of the encryption command

- File listing confirming the creation of the “userdetails_encrypt” file

Encrypting user credentials with AES and Open SSL



The screenshot captures the process of encrypting an plaintext userdetails file using the AES-256-CBC algorithm with OpenSSL. The command openssl enc -aes-256-cbc -salt -in userdetails -out userdetails_encrypt -pass file:./passkey successfully encrypts the file using a randomly generated key stored in passkey. The subsequent ls -l output confirms that the userdetails_encrypt file was created. A warning regarding depreciated key derivation suggests using -pbkdf2 for enhanced security, through this is not required for assignment purposes.

Task 4/4: Secure information using symmetric encryption

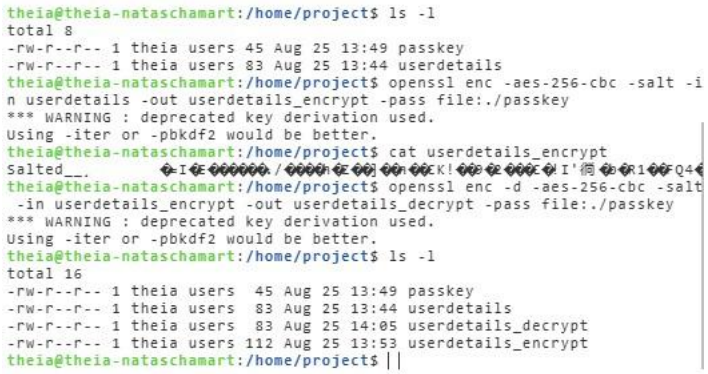
Step 4: Display the contents of the encrypted file using the “cat” command

Prompt: Display the contents of the “userdetails_encrypt” file, even though it is in an unreadable binary format. Ensure the screenshot clearly shows the key detail outlined below.

Key detail:

- Encrypted content of the “userdetails_encrypt” file

Displaying Encrypted File Contents in Binary Format



The screenshot shows that output of the cat userdetails_encrypt command, revealing the binary encrypted contents of the file created using AES-256-CBC encryption and the passkey file. The unreadable characters confirm that the file successfully encrypted and not in plain text.

Task 4/4: Secure information using symmetric encryption

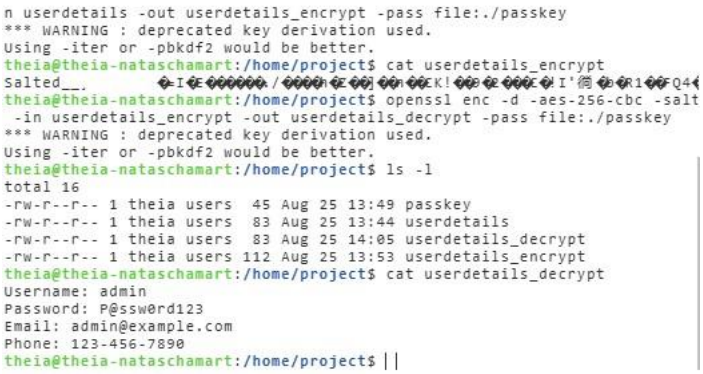
Step 5: Decrypt the “userdetails_encrypt” file using passkey

Prompt: Use the same key to decrypt the previously encrypted file “userdetails_encrypt” and store the output in a new file named “userdetails_decrypt.” Ensure the screenshot clearly shows the key details outlined below.

Key details:

- Successful execution of the “decryption” command
- Creation of the “userdetails_decrypt” file

Decrypting Encrypted User Details Using AES



The screenshots a successful decryption of the userdetails_encrypt file into userdetails_decrypt using OpenSLL with AES-256-CBC. The command executed includes the passkey stored in the passkey file, and th updated file listing confirms creation of the userdetails_decrypt file alongside the original inputs.

Task 4/4: Secure information using symmetric encryption

Step 6: Display the contents of the decrypted file

Prompt: Use a suitable command to display the contents of the “userdetails_decrypt” file and verify that the credentials match those in the original “userdetails” file. Ensure the screenshot clearly shows the key detail outlined below.

Key detail:

- Decrypted contents of the "userdetails_decrypt" file

Displaying Decrypted File Contents with AES Key

```
theia@theia-nataschamart:/home/project$ echo -e "Username: admin\nPasswor
d: P@ssw0rd123\nEmail: admin@example.com\nPhone: 123-456-7890" > userdeta
ils
theia@theia-nataschamart:/home/project$ cat userdetails
Username: admin
Password: P@ssw0rd123
Email: admin@example.com
Phone: 123-456-7890
theia@theia-nataschamart:/home/project$ openssl rand -base64 32 > passkey

theia@theia-nataschamart:/home/project$ ls -l
total 8
-rw-r--r-- 1 theia users 45 Aug 25 13:49 passkey
-rw-r--r-- 1 theia users 83 Aug 25 13:44 userdetails
theia@theia-nataschamart:/home/project$ openssl enc -aes-256-cbc -salt -i
n userdetails -out userdetails_encrypt -pass file:./passkey
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
theia@theia-nataschamart:/home/project$
```

The cat userdetails_decrypt command was executed in the terminal, successfully revealing the decrypted credentials originally stored in the userdetails file. The restored content confirms successful symmetric decryption using AES-256-CBC algorithm and the passkey file, with all fields accurately preserved.