m2_case_study_3

October 11, 2024

1 Module 2 – Sequences and File Operations

1.1 Case Study – 3

LifeTel Telecom needs to implement a secure system for verifying users through a new government-issued Reference ID linked to their fingerprints. The requirement is to encrypt the Reference ID to protect it from hackers, ensuring secure and automated verification during SIM issuance. The system must replace the manual verification process with an efficient, secure, and scalable solution.

[9]:
```
!pip install cryptography
```

```
Requirement already satisfied: cryptography in
c:\users\akram\appdata\local\programs\python\python310\lib\site-packages
(43.0.1)
Requirement already satisfied: cffi>=1.12 in
c:\users\akram\appdata\roaming\python\python310\site-packages (from
cryptography) (1.17.1)
Requirement already satisfied: pycparser in
c:\users\akram\appdata\roaming\python\python310\site-packages (from
cffi>=1.12->cryptography) (2.22)
```

[11]:
```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding
import os

def encrypt_reference_id(reference_id, key):
    """
    Encrypts the Reference ID using AES encryption, returns ciphertext with IV␣
 ↪prepended.
    """
    # Generate a random IV
    iv = os.urandom(16)

    # Create the AES cipher
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),␣
 ↪backend=default_backend())
    encryptor = cipher.encryptor()
```

```
    # Pad the reference ID to match block size
    padder = padding.PKCS7(algorithms.AES.block_size).padder()
    padded_data = padder.update(reference_id.encode()) + padder.finalize()

    # Encrypt the padded data
    ciphertext = encryptor.update(padded_data) + encryptor.finalize()

    # Return the IV concatenated with the ciphertext
    return iv + ciphertext

def decrypt_reference_id(encrypted_data, key):
    """
    Decrypts the data by splitting the IV and ciphertext, then using AES␣
 ↪decryption.
    """
    # Extract the IV and ciphertext
    iv = encrypted_data[:16]
    ciphertext = encrypted_data[16:]

    # Create the AES cipher
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),␣
 ↪backend=default_backend())
    decryptor = cipher.decryptor()

    # Decrypt the ciphertext
    padded_data = decryptor.update(ciphertext) + decryptor.finalize()

    # Remove padding
    unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()
    data = unpadder.update(padded_data) + unpadder.finalize()

    return data.decode()

# Example usage
if __name__ == "__main__":
    # Sample Reference ID
    reference_id = input("Enter 12 Digits ID: ")

    # Generate a 32-byte key for AES-256
    key = os.urandom(32)

    # Encrypt the Reference ID
    encrypted_data = encrypt_reference_id(reference_id, key)
    print(f"Encrypted data: {encrypted_data}")

    # Decrypt the Reference ID
```

```
    decrypted_data = decrypt_reference_id(encrypted_data, key)
    print(f"Decrypted Reference ID: {decrypted_data}")
```

```
Enter 12 Digits ID:  123456789ABC
```

```
Encrypted data: b'\x9f\xc4>\xdb\xca\x90\xe1\x96t*\xce\xdf\xd8\xf3|\xdd9.\xd9\\\x
a9\x88\x16m%\x0e\xd2\xde\xd7\x011\x9a'
Decrypted Reference ID: 123456789ABC
```

1.2 Encryption for Securing Reference IDs in the Telecom System

For securing the Reference ID in the telecom system, AES (Advanced Encryption Standard) is recommended as the encryption algorithm. AES is a highly secure, efficient, and standardized encryption method, making it an excellent choice for safeguarding sensitive data like the Reference ID.

1.2.1 Why AES?

Security: AES is a widely trusted symmetric encryption algorithm, used by governments and industries globally. It offers a high level of security for sensitive data. Efficiency: AES can encrypt and decrypt large volumes of data quickly, making it well-suited for telecom systems that require scalability as user volume increases. Standardization: AES is an industry-standard encryption method, which ensures compatibility across systems and provides confidence in its long-standing security properties.

1.2.2 Key Aspects of the Encryption Process:

AES Encryption: The function encrypt_reference_id() encrypts the Reference ID using AES in CBC mode. It generates a unique Initialization Vector (IV) for each encryption process to enhance security.

1.2.3 Padding:

Since AES operates on fixed block sizes (16 bytes), if the Reference ID is shorter than the block size, padding is applied using the PKCS7 padding scheme to ensure it fits properly into the encryption process.

1.2.4 Key Management:

The encryption key is a critical component of the system's security. Proper key management is essential. In a production environment, keys should be securely stored and managed using systems such as a Key Management Service (KMS).

1.2.5 Decryption:

The function decrypt_reference_id() handles decryption by reversing the encryption process. It decrypts the data and removes any padding to retrieve the original Reference ID.

1.2.6 Why Use an Initialization Vector (IV)?

The Initialization Vector (IV) plays a crucial role in AES encryption by ensuring that even if the same Reference ID is encrypted multiple times, the resulting ciphertext will be different each time. This enhances confidentiality by preventing attackers from recognizing patterns. While the IV is randomly generated for each encryption, it does not need to be kept secret and can be safely stored alongside the ciphertext.

1.2.7 Security Considerations:

Secure Key Storage: The entire security of the encryption process relies on keeping the encryption key secure. Best practices for key storage, such as using encrypted databases or hardware security modules (HSM), are essential to avoid key compromise.

IV Management: For each encryption, a new IV must be generated to prevent ciphertext reuse. The IV should be stored or transmitted alongside the ciphertext, as it is required during the decryption process.

1.2.8 Next Steps for Integration:

To fully integrate this encryption process into the telecom system:

Incorporate Encryption in User Registration: Implement the encryption process during user registration and verification to securely handle Reference IDs.

Utilize Key Management Solutions: Consider deploying a secure Key Management Solution (KMS) to manage the encryption keys, ensuring that keys are handled safely and rotated as needed.

Storing the IV Along with the Ciphertext: In practical implementations, the IV is often stored or transmitted with the ciphertext, as it is necessary for decryption. The IV can be prepended to the ciphertext, allowing it to be easily extracted and used during decryption. This ensures that even though the IV changes for each encryption, decryption can still be done correctly with the stored IV and the encryption key.

[6]:
```
#### Mr Akram M'Tir 10-10-2024
```

[ ]:
```

```