Bài tập 3: Cài đặt CBC mode và CTR mode

Nguyễn Hàn My – 20205215

Yêu cầu

In this project, you will implement two encryption/decryption systems, one using AES in CBC mode and another using AES in counter mode (CTR). In both cases, the 16-byte encryption IV is chosen at random and is prepended to the ciphertext.

For CBC encryption we use the PKCS5 padding scheme discussed in the lecture. While we ask that you implement both encryption and decryption, we will only test the decryption function. In the following questions you are given an AES key and a ciphertext (both are hex encoded) and your goal is to recover the plaintext.

For an implementation of AES, you may use an existing crypto library such as PyCrypto (Python), Crypto++ (C++), or any other. While using the built-in AES functions is fine, we ask that as a learning experience, you implement CBC and CTR modes yourself.

You must submit both the source code of the program you wrote to decode and a report describing how to decrypt.

Trả lời

- Question 1
  Plaintext: Basic CBC mode encryption needs padding.

- Question 2
  Plaintext: Our implementation uses rand. IV

- Question 3
  Plaintext: CTR mode lets you build a stream cipher from a block cipher.

- Question 4
  Plaintext: Always avoid the two time pad!

Giải thích

Thư viện

- **aes**: mã hóa AES
- rand: tạo số ngẫu nhiên
- hex: mã hóa/giải mã hexa

Các thuật toán

Giải mã CBC

1. Đầu tiên, chúng ta cần có một khối IV (Initialization Vector) đã được sử dụng trong quá trình mã hóa.
2. Khối ciphertext đầu tiên (ciphertext block thứ 1) được giải mã bằng cách sử dụng khóa mã hóa AES.
3. Kết quả giải mã được XOR với khối ciphertext thứ hai (ciphertext block thứ 2) để tạo ra plaintext block thứ nhất.
4. Quá trình này được lặp lại với các khối ciphertext tiếp theo, trong đó plaintext block trước đó được XOR với khối ciphertext hiện tại để tạo ra plaintext block mới.
5. Quá trình lặp lại cho đến khi giải mã tất cả các khối ciphertext để thu được plaintext ban đầu.

Mã hóa CBC

1. Đầu tiên, chúng ta cần có một khối IV (Initialization Vector) để khởi tạo quá trình mã hóa.
2. Plaintext ban đầu được chia thành các khối có cùng độ dài.
3. Khối plaintext đầu tiên được XOR với khối IV.
4. Kết quả XOR được mã hóa bằng cách sử dụng khóa mã hóa AES, tạo ra khối ciphertext đầu tiên.
5. Khối ciphertext đầu tiên này được sử dụng để XOR với khối plaintext thứ hai để tạo ra khối ciphertext thứ hai.
6. Quá trình này được lặp lại với các khối plaintext tiếp theo, trong đó khối ciphertext trước đó được XOR với khối plaintext hiện tại để tạo ra khối ciphertext mới.
7. Quá trình lặp lại cho đến khi mã hóa tất cả các khối plaintext để thu được các khối ciphertext tương ứng.

Giải mã CTR

1. Đầu tiên, chúng ta cần có một khối nonce (number used once) đã được sử dụng trong quá trình mã hóa.
2. Khối ciphertext được chia thành các khối có cùng độ dài.
3. Counter ban đầu được khởi tạo với giá trị từ nonce.
4. Khối counter được mã hóa bằng cách sử dụng khóa mã hóa AES, tạo ra một khối keystream.
5. Khối keystream được XOR với khối ciphertext để tạo ra khối plaintext.
6. Counter được tăng lên một giá trị để chuẩn bị cho việc mã hóa khối tiếp theo.
7. Quá trình này được lặp lại với các khối ciphertext tiếp theo để giải mã tất cả các khối ciphertext và thu được plaintext ban đầu.

Mã hóa CTR

1. Đầu tiên, chúng ta cần có một khối nonce (number used once) để khởi tạo quá trình mã hóa.
2. Plaintext ban đầu được chia thành các khối có cùng độ dài.
3. Counter ban đầu được khởi tạo với giá trị từ nonce.
4. Khối counter được mã hóa bằng cách sử dụng khóa mã hóa AES, tạo ra một khối keystream.
5. Khối keystream được XOR với khối plaintext để tạo ra khối ciphertext.
6. Counter được tăng lên một giá trị để chuẩn bị cho việc mã hóa khối tiếp theo.
7. Quá trình này được lặp lại với các khối plaintext tiếp theo để mã hóa tất cả các khối plaintext và thu được các khối ciphertext tương ứng.

Hàm main()

- Khóa (`cbc_key`, `ctr_key`) và ciphertexts (`ciphertext_1`, `ciphertext_2`, `ciphertext_3`, `ciphertext_4`) được định nghĩa cho việc kiểm tra.
- Các hàm `test_decrypt()` được gọi để kiểm tra việc giải mã các ciphertext đã cho.
- Asserts được sử dụng để kiểm tra tính đúng đắn của mã nguồn.