Тамара Георгиева 213207

Документација

1. Внесување на библиотеки и функции

- from Crypto.Cipher import AES: Ова овозможува користење на AES алгоритмот за шифрирање и декрипција од библиотеката Crypto.
- import Crypto.Random: Овозможува користење на функции за генерирање на случајни бајтови од библиотеката Crypto.
- import math: Внесување на модулот за математички операции

2. generate_counter_iv(counter): Оваа функција генерира и враќа иницијален вектор

- iv = counter.to_bytes(16, byteorder='big'):
- return iv

3. calculate_mic(encrypted_data, key, counter): Оваа функција пресметува MIC

- mic = b"": Иницијализација на променлива за зачувување на MIC-от
- num_blocks = math.ceil(len(data) / 16): Пресметување на бројот на блокови
- for i in range(num_blocks): Циклус за обработка на секој блок
- counter_iv = generate_counter_iv(counter + i): Генерирање на иницијалниот вектор за моменталниот блок
- keystream_block = AES.new(key, AES.MODE_ECB).encrypt(counter_iv): Шифрирање на иницијалниот вектор за да се добие keystream блок.
- mic_block= bytearray(16): Иницијализација на бајтова низа за податочниот блок.
- Следува циклус за XOR операција помеѓу енкриптиранте податоци и keystream блокот:
- mic_block[j] = encrypted_data[index] ^ keystream_block[j]: Извршување на XOR операција
- mic += mic_block: Додавање на резултантниот MIC блок

- return mic: Враќање на MIC

4. pad_data(data): Оваа функција додава падинг на податоците

- remaining_bytes = 16 len(data) % 16: Пресметување на бројот на бајтови потребни за падинг.
- padding = bytes([remaining_bytes]) * remaining_bytes: Генерирање на падинг
- return data + padding: Додавање на падингот на податоците и враќање на резултатот

5. unpad_data(data): Оваа функција го отстранува падингот од податоците.

- padding_length = data[1]: Читање на последниот бајт од податоците
- return data[:padding_length]: Враќање на податоците без падингот.

Тамара Георгиева 213207

6. encrypt_data_ctr(data, key, counter): Оваа функција е одговорна за енкрипција на податоците

- cipher = AES.new(key, AES.MODE_ECB): Креирање на објект за шифрирање со AES клуч
- encrypted_data = b"": Иницијализација на променлива за чување на енкриптираните податоци.
- num_blocks = len(data) // 16: Пресметување на бројот на податочни блокови.
- for i in range(num_blocks): Циклус за обработување на секој блок.
- counter_iv = generate_counter_iv(counter + i): Генерирање на иницијален вектор за моменталниот блок
- keystream_block = cipher.encrypt(counter_iv): Шифрирање на иницијалниот вектор за да се добие keystream блок.
- data_block = data[i * 16: (i + 1) * 16]: Избирање на моменталниот блок од податоците за шифрирање.
- encrypted_block = bytes(data_byte ^ keystream_byte for data_byte, keystream_byte in zip(data_block, keystream_block)): Извршување на XOR операција
- encrypted_data += encrypted_block: Додавање на енкриптираниот блок

- return encrypted_data: Враќање на енкриптираните податоци.

7. simulate_send_receive(encrypted_data, key, counter, original_data_mic): Оваа функција симулира пренос и прием на енкриптирани податоци и проверка на MIC-от

- received_encrypted_data = encrypted_data: Симулирање на примањето на енкриптираните податоци.
- decrypted_data = encrypt_data_ctr(received_encrypted_data, key, counter): Декриптирање на примените податоци
- unpadded_data = unpad_data(decrypted_data): Отстранување на падингот од декриптираните податоци.
- decrypted_data_mic = calculate_mic(unpadded_data, key, counter): Пресметување на MIC-от на декриптираните податоци.
- if original_data_mic != decrypted_data_mic: Проверка дали MIC-от на примените податоци е ист како и оригиналниот.
- raise Exception("MIC-от не се совпаѓа!"): Ако MIC-от не се совпаѓа, се фрла грешка.
- return unpadded_data: Враќање на декриптираните податоци без падинг.