Lab 2

Hrishikesh Bingewar 24b3983

September 2025

1 Quesition 1: OpenSSL Decryption

The objective of this task was to decrypt a given ciphertext that was encrypted using the AES-128 algorithm in Cipher Block Chaining (CBC) mode. We were provided with:

A 128-bit key (in hexadecimal format),
An Initialization Vector (IV) (in hexadecimal format), and
A ciphertext (stored in a binary file).

1.1 Theory

AES(Advanced Encryption Standard) is a symmetric block cipher that encrypts data in fixed-size blocks of 128 bits using keys of length 128, 192, or 256 bits. In this experiment, we use a 128-bit key.

CBC (Cipher Block Chaining) mode introduces randomness by XORing each plaintext block with the previous ciphertext block before encryption.The first block is XORed with an Initialization Vector (IV).

1.2 Implementation

During decryption, each ciphertext block is first decrypted and then XORed with the previous ciphertext block (or IV for the first block) to recover plaintext. Steps:

- Hex decoding: The key and IV provided as hex strings are converted to raw bytes (bytes.fromhex).
- Cipher initialization: The AES cipher object is created in CBC mode with the given key and IV.
- Decryption: The ciphertext is decrypted into padded plaintext.
- Padding removal: The last byte indicates how many padding bytes were added, which are then removed.
- Plaintext recovery: The result is displayed either as UTF-8 text (if printable) or raw bytes.

2 Question 2

In this challenge, we exploit the redundancy of ECB to obtain the flag.

2.1 Solution

In AES-ECB, encryption is done independently for each 16-byte block. This means that patterns in plaintext are preserved in ciphertext.
If part of the plaintext is known, it can be used to build a codebook mapping ciphertext blocks to characters, and we know the plaintext for the header and its ciphertext.

This codebook can then be reused to decode the rest of the message

2.2 Implementation

The program works as follows:

- We read the ciphertext from file and split into 16-byte blocks.
- Build a mapping between ciphertext blocks and characters using the known header text.
- Decode remaining ciphertext by checking if blocks match any in the table.
- If a match exists, the corresponding plaintext character is recovered. Otherwise, the character is marked as unknown (?).
- Save output to a file and display unmapped blocks.

3 Question 3- The catastrophic equality

In this problem, the encryption is carried out using one key as both IV and the encrytption key in CBC mode of encryption. We exploit this to find the flag.

3.1 Implementation

- We first submit a harmless message "a=s" via Choice 1, and obtain its ciphertext.
- Then, we construct input as $C1||0\hat{B}LOCK||C1$ and send it to the server with Choice 2. On decryption, this leaks information because of CBC structure and the reused IV = key. The server prints back the error along with the decrypted text (partially).
- From the decryption output, we can isolate two plaintext blocks (P1 and P3).
  P1 = Dec_k(C1) $\oplus k$
  $P3 = Dec\_k(C1) \oplus 0$
  $k = P1 \oplus P3$
- With the key k, we locally encrypt the padded plaintext "admin=true" under AES-CBC with IV = k. This produces a valid ciphertext that the server accepts.
- When the server recognizes us as admin, it sends back the encrypted flag. Using the recovered key, we decrypt it locally to obtain the plaintext flag.

4 Question 4- Never painted by the numbers

We interact with a server that behaves like an echo server: Whatever string we input, the server encrypts and outputs it back. Special case: if we input !flag, the server outputs the encrypted flag instead. Encryption is done using AES in CTR mode with a secret key.

4.1 Implementation

- We send a very long dummy string of null bytes (00 * 1600). The encrypted output = pure keystream.
- Send the input !flag and capture the encrypted flag bytes.
- Compute how many blocks the flag starts at and slice the keystream from the dummy output to match the flag's ciphertext length.
- XOR the encrypted flag with the keystream slice: P $\oplus KS. This yields the plaintext flag$.