Birzeit University
Faculty of Engineering and Technology
Department of Electrical and Computer Engineering
ENCS4320 – Applied Cryptography (Term 1243)
Task # 2 *(Symmetric Crypto Systems: Implementation and Analysis)* – Due Tuesday, August 12, 2025

In this task, you are required to implement a complete **AES-128 encryption and decryption** program from scratch, following the algorithm step-by-step as discussed in class. The implementation must also support the **Cipher Block Chaining (CBC)** mode of operation.

Instructions:

A) Core AES Implementation:

- You must implement each **major step of the AES algorithm** as a separate function:
  - *Round Function*: *SubBytes/InvSubBytes*, *ShiftRows/InvShiftRows*, *MixColumns/InvMixColumns*, *AddRoundKey*.
  - *Key Expansion* (Round Key Generation)
- All AES operations (*SubBytes*, *MixColumns*, etc.) should be implemented **algorithmically**:
  - **Avoid using lookup tables** (e.g., precomputed S-boxes or Galois multiplication tables).
  - If lookup tables are used, **clearly document their use** in your report; note that this will result in a **deduction of up to 20%** of the task grade.

B) CBC Mode Integration:

- Implement CBC-mode encryption and decryption using your AES-128 functions.
- Support handling plaintexts of arbitrary length.
- Use a **128-bit Initialization Vector (IV)** for CBC mode (prompt the user to enter it).

Structure and File Organization:

A) AES Module:

- Place all AES-related functions and logic in a single file named: ***task2_aes*** (e.g., ***task2_aes.py***)
- This file should **not** contain any code for user input, output, or interaction.

B) Runner Script:

- In a separate script (e.g., ***task2_run_aes.py***), create an interactive console program that:
  - Prompts the user to select the operation: **Encrypt (E)** or **Decrypt (D)**.
  - Prompts the user to enter:
    - A **128-bit plaintext or ciphertext** (in **hexadecimal format**).
    - A **128-bit AES key** (in **hexadecimal format**).
    - A **128-bit IV** (in **hexadecimal format**).
  - Calls AES-CBC encryption or decryption function.
  - Displays the resulting ciphertext or plaintext in hexadecimal format.

C) Avalanche Effect Script:

- Create a third script named: ***task2_aes_avalanche_analysis***.
- This script should:
  - Import your AES functions from ***task2_aes***.
  - Choose a random 128-bit plaintext **P_1**, a random 128-bit key **K_1**, and a random IV.
  - Compute **C_1 = AES_CBC_encrypt(K_1, P_1, IV)**.
  - Perform two experiments:
    a) **Plaintext Bit Flip:** Flip **one random bit** in **P_1** → get **P_1'**, compute **C_2 = AES_CBC_encrypt(K_1, P_1', IV)**.
    b) **Key Bit Flip:** Flip **one random bit** in **K_1** → get **K_1'**, compute **C_2 = AES_CBC_encrypt(K_1', P_1, IV)**.
  - Repeat both experiments **10 times**.
  - Display a summary table showing how many bits differed between **C_1** and **C_2** in each experiment.
- Comment on the observed avalanche effect.

D) Extended Analysis:

- Perform the following analyses:

a) Bit Error in Ciphertext:

- Encrypt a plaintext using AES-CBC.
- Flip a single bit in a random position in the resulting ciphertext.
- Decrypt the modified ciphertext and display the resulting plaintext.
- Analyze and describe:
  - Which blocks are affected in the decrypted message?
  - How many plaintext blocks are corrupted as a result?
  - Why does this behavior occur in CBC mode?

b) Loss of a Ciphertext Block:

- Encrypt a multi-block plaintext using AES-CBC.
- Simulate the loss of one full ciphertext block during transmission.
- Decrypt the incomplete ciphertext and display the resulting plaintext.
- Analyze and describe:
  - Which blocks are affected in the decrypted output?
  - Can any block still be decrypted correctly?
  - What does this reveal about error propagation in CBC mode?

c) Data Exposure in Ciphertext:

- Encrypt a black-and-white image (e.g., BMP, PNG format) using AES-CBC.
- Simulate an attacker (Trudy) sniffing the ciphertext during transmission.
- Attempt to visualize the ciphertext as an image (e.g., by treating the ciphertext as raw grayscale pixel data).
- Analyze and describe:
  - What does the reconstructed image look like?
  - Can any recognizable patterns be seen?
  - What does this imply about CBC mode's resistance to data leakage compared to ECB?

Deliverables:

1) ***Source code*** files (e.g., ***task2_aes.py***, ***task2_run_aes.py***, and ***task2_aes_avalanche_analysis.py***).
2) A ***brief documentation*** that includes:
   - Overview of your implementation.
   - Sample input/output for encryption and decryption.
   - Avalanche effect results and interpretation.
   - Error and data exposure analysis with observations.
   - Any assumptions made.
   - A clear note if any **lookup tables** (e.g., *S-box*, *MixColumns*) were used.

GOOD LUCK