CS454/654 Reliability and Security of Computing Systems
Fall 2024
Homework 2
Total points: 100

Objective: Advanced Encryption Standard (AES)

- **Programming Language:** Use a programming language of your choice (e.g., Python, Java, C++).
- **Code Submission:** Submit all source code files with proper filenames and extensions.
- **Documentation:** Include comments in your code to explain your logic.
- **Report:** Provide a brief report explaining your implementation, the challenges faced, and how you tested your code.

NB: When designing the algorithm, consider that the AES input size is 128 bits, and the key size is also 128 bits. **Don't use any external library to implement your solution**. For AES S-boxes, Inverse S-boxes, and other matrixes required for calculation, please refer to the slide **Lect-12-CH-6.pptx** or the book.

Question 1: Implement the SubBytes Transformation (15 marks)

Task: Write a function to perform the SubBytes transformation on a 4×4 state matrix.

Requirements: The function should:

- Accept a 4×4 matrix representing the state.
- Use the AES S-box to substitute each byte in the state.
- Return the transformed state matrix.

Question 2: Implement the ShiftRows Transformation (10 marks)

Task: Write a function to perform the **ShiftRows** transformation on a 4×4 state matrix.

Requirements: The function should:

- Accept a 4×4 state matrix as input.
- Shift the rows as per AES specifications.
- Return the transformed state matrix.

Question 3: Implement the MixColumns Transformation (15 marks)

Task: Write a function to perform the **MixColumns** transformation on a 4×4 state matrix.

Requirements: The function should:

- Accept a 4×4 state matrix as input.
- Multiply each column of the state by a fixed polynomial matrix in the Galois Field (GF(2^8)).
- Return the transformed state matrix.

Question 4: Implement the AddRoundKey Transformation (10 marks)

Task: Write a function to perform the **AddRoundKey** transformation.

Requirements: The function should:

- Accept two 4×4 matrices: the state matrix and the round key matrix.
- Perform a bitwise XOR between the state and the round key.
- Return the resulting state matrix.

Question 5: Implement the Key Expansion Function (15 marks)

Task: Write a function to perform the AES **Key Expansion** for 128-bit keys.

Requirements: The function should:

- Accept a 16-byte cipher key.
- Generate all round keys needed for AES encryption (10 rounds for 128-bit keys).
- Implement the necessary operations:
  - **RotWord:** Rotate a word (4 bytes) left by one byte.
  - **SubWord:** Apply the SubBytes transformation to each byte of a word.
  - **Rcon:** Incorporate the round constant.
- Return a list of round keys.

Question 6: Assemble the AES Encryption Algorithm (10 marks)

Task: Write a function that combines all the previous functions to perform AES encryption on a single 16-byte (128-bit) block.

Requirements: The function should:

- Accept a 16-byte plaintext block and a 16-byte cipher key.
- Use your **Key Expansion** function to generate round keys.
- Perform the initial AddRoundKey.
- Execute nine rounds consisting of SubBytes, ShiftRows, MixColumns, and AddRoundKey.
- Execute the final round of SubBytes, ShiftRows, and AddRoundKey (without MixColumns).
- Return the 16-byte ciphertext block.

Question 7: Testing and Verification (10 marks)

Task: Test your AES encryption function with the provided test vectors and verify the correctness.

Requirements:

- Use the example output:

Plaintext:	`0123456789abcdeffedcba9876543210`
Key:	`0f1571c947d9e8590cb7add6af7f6798`
Ciphertext:	`ff0b844a0853bf7c6934ab4364148fb9`

- Generate your unique test cases and check the outputs

Question 8: Demonstrate the Avalanche Effect in AES (15 marks)

Task: Extend your AES implementation to demonstrate the avalanche effect by showing how a one-bit change in the plaintext or key affects the ciphertext across each round.

Requirements:

- Choose one of the following:

    - Option 1: Change a single bit in the plaintext while keeping the key constant.
    - Option 2: Change a single bit in the key while keeping the plaintext constant.
- For each round of AES encryption:
    - Record the state of the ciphertext after each round for both the original and the modified input.
    - Calculate and display the number of differing bits between the original and modified ciphertexts after each round.
- Include in your report:
    - A detailed explanation of your methodology.
    - A table showing the number of differing bits at each round.

Example:

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Bonus Question: Implement AES Decryption (10 marks)

Task: Write a function to implement the AES decryption process for a 16-byte ciphertext block.

Requirements: The function should:

- Accept a 16-byte ciphertext block and a 16-byte key.
- Use your Key Expansion function to generate round keys.
- Perform the decryption steps, which are the inverse of the encryption process:
- Inverse AddRoundKey
- Inverse ShiftRows
- Inverse SubBytes
- Inverse MixColumns (for rounds except the final one)
- Return the original plaintext block.

Submission Guidelines

- **Code Files:**
  - Submit all source code files (.py, .java, .cpp, etc.).
  - Ensure the code is well-organized and has appropriate filenames.
  - Keep your implemented functions as modular as possible so that individual functions can be tested independently
- **Report Document:**
  - A brief report in PDF format.
  - Include:
    - Overview of your implementation for each question.
    - Any challenges faced and solutions.
    - Testing methodology and results.
    - Screenshots of your output logs.
- **Compression:**
  - Compress all files into a single ZIP file named: AES_Assignment_[YourName].zip