Experiment 2
Information Security: Encryption/Decryption

Part 1:

1. Caesar Cipher: The C code exp2.c has been compiled with gcc compiler with the command gcc exp2.c -o encrypt_decrypt. The code has been run with the key input range -26 to 26. The plain text should have all English capital letters.

2. Advanced Encryption Standard (AES): The given codes have been compiled and observed the output according to the given instruction.

Caeser Cipher : The output:

```
Positive interger makes the left shift and negative integer will make right shift.
The key value has the range -26 to 26.
Please enter the key:
5
The plain text should contain only capiital letters
Please give the plain text
PANKAJ
The cypher text:
UFTQFO
```

AES: The screenshot for AES step 5 is given below.

```
Plain Text Before Encryption TXT  : 'gainesvillekjhkjhlkjlkjnkjnkjnlklkmlkmn'
Plain Text Before Encryption HEX  : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x6b 0x6a 0x68 0x6b 0
x6a
Cipher Text After Encryption      : '
Cipher Text After Encryption HEX  : 0x83 0xf9 0x19 0x2d 0x7a 0xcc 0x6b 0xee 0x5b 0x85 0x28 0x16 0x1a 0xe 0xa1 0x
f1
Plain Text After Decryption       : 'gainesvillekjhkj'
Plain Text After Decryption HEX   : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x6b 0x6a 0x68 0x6b 0
x6a
```

AES: The screenshot for AES step 6 is given below

```
Plain Text Before Encryption TXT  : 'gainesville'
Plain Text Before Encryption HEX  : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x0 0x0 0x0 0x0 0x0
Cipher Text After Encryption      : '
Cipher Text After Encryption HEX  : 0x1a 0x1e 0x6d 0x9d 0x4c 0x5f 0xf7 0x32 0x27 0x84 0x65 0xbd 0x43 0x2b 0xa2 0
x7e
Plain Text After Decryption       : 'gainesville'
Plain Text After Decryption HEX   : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x0 0x0 0x0 0x0 0x0
```

AES: The screenshot for AES step 9 is given below:

```
Plain Text Before Encryption TXT  : 'universityoffloridauniversityofflorida'
Plain Text Before Encryption HEX  : 0x75 0x6e 0x69 0x76 0x65 0x72 0x73 0x69 0x74 0x79 0x6f 0x66 0x66 0x6c 0x6f 0
x72
Cipher Text After Encryption      : '
Cipher Text After Encryption HEX  : 0x4a 0x7 0x9e 0x8b 0x34 0x3b 0xc1 0x25 0xcb 0xe3 0x61 0xa3 0x6c 0x1a 0x14 0x
c4
Plain Text After Decryption       : 'universityofflor'
Plain Text After Decryption HEX   : 0x75 0x6e 0x69 0x76 0x65 0x72 0x73 0x69 0x74 0x79 0x6f 0x66 0x66 0x6c 0x6f 0
x72
```

Experiment 2 Extension

I extended this program as follows with the attached source code in the repository:

“Modify the AES code for any size of message length. Turn in your code and screenshots when executed”

1. **Padding Implementation**: A `padData` function was introduced to add PKCS#7 padding to the input data so that its length becomes a multiple of the AES block size (16 bytes). This function fills the extra space with bytes, each of which is equal to the number of padding bytes added.

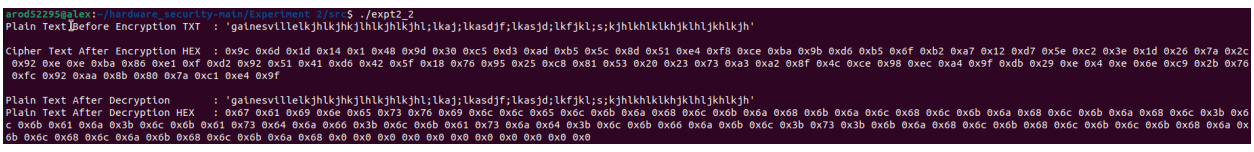
2. **Unpadding Implementation**: An `unpadData` function was created to remove the padding after decryption. This function looks at the last byte of the decrypted data to determine the number of padding bytes to remove, ensuring the original data is restored.

3. **Buffer Size Adjustment**: The buffer sizes in the `aes_encrypt` and `aes_decrypt` functions were adjusted to accommodate the padded data. This means that the output buffer for encryption must be large enough to hold the original data plus up to 16 extra bytes of padding, and the buffer for decryption must be at least as large as the encrypted data.

4. **Function Signature Change**: The signatures of the `aes_encrypt` and `aes_decrypt` functions were modified to accept the length of the input data and an output buffer to store the results. This change was reflected in both the function declarations (in the header file) and definitions (in the implementation file).

5. **Main Function Update**: In the main program, the `aes_encrypt` and `aes_decrypt` functions were called with the correct arguments, including the input data, its length, the output buffer, and the encryption key.

6. **Compilation and Execution**: The program was compiled with the updated `aes_encrypt` and `aes_decrypt` functions, and executed to ensure that data of any length could be encrypted and decrypted correctly.



Summary:
This hardware security lesson focuses on two encryption techniques: the Caesar Cipher and the

Advanced Encryption Standard (AES). The Caesar Cipher is a simple substitution cipher where each letter in the plaintext is shifted a certain number of positions up or down the alphabet. The lesson includes a programming the Caesar Cipher in C, emphasizing case sensitivity and ignoring non-letter characters.

The second part of the lesson provides information on AES, a more sophisticated encryption method based on a substitution-permutation network. Unlike its predecessor DES, AES operates on a fixed block size of 128 bits and allows for key sizes of 128, 192, or 256 bits. The practical exercises involve compiling and running AES encryption code, modifying input messages, and analyzing the resultant ciphertext.