CS 4920/5920 Spring 2022

HW 4

This HW is due on 4/8. Explain how you reached your answers. Answers without explanations will receive no to little credit. Submit a zip file including your answer sheets and the following for the programming/implementation problem(s): code, compiler outputs/executables if applicable, and the programming output (what the problems are asking for).

Problem 1.

Investigate Simplified AES (S-AES) based on GF(2^4). S-AES is for instructional purpose and is a simpler version of AES using smaller numbers.

a. What is the block length and the structure of the state array? Express them in bits.
b. What is the modulus/prime polynomial used for the GF(2^4) operations in S-AES?
c. Show that the matrix given in the following, with entries in GF(2^4), is the inverse of the matrix used in the MixColumns step of S-AES.

$$\begin{pmatrix} x^3+1 & x \\ x & x^3+1 \end{pmatrix}$$

Problem 2.

Let's implement AES.

a. Perform Key Expansion using the Key {0e0071c947d9e8591cb7add6af7f6798} and construct and output a table like Table 6.3 in the textbook. Have the table the same structure/format as Table 6.3 and include the entry values.
b. Perform AES on Plaintext {1321456789abcdeffedcba9876543210} using the key in the previous part and construct and output a table like Table 6.4. Have the table the same structure/format as Table 6.4 and include the entry values.
c. Repeat the previous two parts with a key and a plaintext generated by yourself. Clearly identify the key and the plaintext you generated and are using for this problem.

Let's study the Avalanche Effect in AES.

d. Explain the Avalanche Effect and why it is a desirable feature.
e. Given the Plaintext *P*= {1200456789abcdeffedcba9876543210} as the baseline, generate two more plaintexts, each of which are different from *P* by one bit. For each of the plaintext you generated, perform AES on them using the key {0f0071c947d9e8591cb7add6af7f6798}. To compare each of the new plaintexts and *P*, construct two tables like Table 6.5 in the textbook. Have the table the same structure/format as Table 6.5 and include the entry values.

Problem 3.

The following questions are about the CBC mode.

a. Is it possible to perform encryption operations in parallel on multiple blocks of plaintext in CBC mode? How about decryption?
b. In the CBC mode, an error in a block of transmitted ciphertext propagates. For example, an error in the transmitted C_1 (Figure 7.4) corrupts P_1 and P_2. Are any blocks beyond P_2 affected?

c. Suppose that there is a bit error in the source/transmitter version of P_1 in the CBC mode. Through how many ciphertext blocks is this error propagated at the source transmitter? What is the effect at the receiver?

Now let's repeat the questions for the CTR mode.

d. Is it possible to perform encryption operations in parallel on multiple blocks of plaintext in CTR mode? How about decryption?
e. In the CTR mode, an error in a block of transmitted ciphertext occurs. Are any blocks beyond P_2 affected?
f. Suppose that there is a bit error in the source/transmitter version of P_1 in the CTR mode. Through how many ciphertext blocks is this error propagated at the source transmitter? What is the effect at the receiver?

Problem 4.

Suppose you have a true random bit generator where each bit in the generated stream has the same probability of being a 0 or 1 as any other bit in the stream and that the bits are independent, i.e., the bits are generated from identical independent distribution (IID). However, the bit stream is biased. The probability of a 1 is $0.5 + \partial$ and the probability of a 0 is $0.5 - \partial$, where $0 < \partial < 0.5$. A simple conditioning algorithm is as follows: examine the bit stream as a sequence of nonoverlapping pairs. Discard all 00 and 11 pairs. Replace each 01 pair with 0 and each 10 pair with 1.

a. What is the probability of occurrence of each pair in the original sequence?
b. What is the probability of occurrence of 0 and 1 in the modified sequence?
c. What is the expected number of input bits to produce x output bits?
d. Suppose that the algorithm uses overlapping successive bit pairs instead of non-overlapping successive bit pairs. That is, the first output bit is based on input bits 1 and 2, the second output bit is based on input bits 2 and 3, and so on. What can you say about the output bit stream?

Another approach to conditioning is to consider the bit stream as a sequence of non-overlapping groups of n bits each and output the parity of each group. That is, if a group contains an odd number of ones, the output is 1; otherwise the output is 0.

e. Express this operation in terms of a basic Boolean function.
f. Assume, as in before, that the probability of a 1 is $0.5 + \partial$. If each group consists of 2 bits, what is the probability of an output of 1?
g. If each group consists of 4 bits, what is the probability of an output of 1?
h. Generalize the result to find the probability of an output of 1 for input groups of n bits.

Problem 5.

a. State and prove Fermat's Theorem.
b. State and prove Euler's Theorem.
a. Prove that $\phi(n)$ is even for $n > 2$ where $\phi(n)$ is the Euler's Totient Function.

Problem 6.

a. Find all primitive roots of 13. Construct a table like Table 2.7 in the textbook to find the primitive roots.
b. Given 2 as a primitive root of 29 (base 2, modulo 29), construct a table of discrete logarithms like Table 2.8 in the textbook.