Cryptography and Network Security (CS60065)

AUTUMN, 2021-2022

Home Work

QUESTION : 1 (The Feistel cipher)

Consider a Feistel cipher composed of 16 rounds with block length 128 bits and key length 128 bits. Suppose that, for a given k, the key scheduling algorithm determines values for the first 8 round keys, k1, k2, ..., k8, and then sets k9 = k8, k10 = k7, k11 = k6, ..., k16 = k1.

Suppose you have a ciphertext c. Explain how, with access to an encryption oracle, you can decrypt c and determine m using just a single oracle query. This shows that such a cipher is vulnerable to a chosen plaintext attack.

QUESTION : 2 (The SubByte Value)

Calculate the SubByte value of $(53)_{16}$

QUESTION : 3 (Euclidean Algorithm)

Determine gcd(24140, 16762) by using Euclidean Algorithm.

QUESTION : 4 (Euclidean Algorithm)

Using the extended Euclidean algorithm, find the multiplicative inverse of 24140 mod 40902

QUESTION : 5 (Field Arithmetic)

For polynomial arithmetic with coefficients in Z_{10}, perform the calculation:
$(6x^2 + x + 3) \text{ x } (5x^2 + 2)$

QUESTION : 6 (Field Arithmetic)

Develop a generator table for $GF(2^4)$ with $m(x) = x^4 + x + 1$.

[A generator g of a finite field F of order q (contains q elements) is an element whose first q - 1 powers generate all the nonzero elements of F. That is, the elements of F consist of 0, g^0 , g^1 , … , g^{q-2} . Consider a field F defined by a polynomial f(x). An element b contained in F is called a root of the polynomial if f(b) = 0. Specifically a root g of an irreducible polynomial is a generator of the finite field defined on that polynomial. For this generator table define the finite field of the form $GF(2^4)$, using the irreducible polynomial with the power representation, as well as the polynomial and binary representations.]

QUESTION : 7 (Related to AES)

Show that $x^i (x^4 + 1) = x^i \bmod 4$.

QUESTION : 8 (Related to AES)

Compute the output of the MixColumns transformation for the following sequence of input bytes "67 89 AB CD". Apply the InvMixColumns transformation to the obtained result to verify your calculations. Change the first byte of the input from '67' to '77', perform the MixColumns transformation again for the new input, and determine how many bits have changed in the output.