**Homework #3 Deneme**

Due date: 25 November 2020, by 11:59 pm.

**Notes**:

* For Question 1, you can use a Python module for arithmetic in GF(28).
* For Question 2, you can use “**lfsrf.py**” provided to you earlier.
* You are expected to submit your answer document as well as two Python codes for Questions 1 and 2, respectively.
* Zip your programs and add a readme.txt document (if necessary) to explain the programs and how to use them.
* Name your winzip file as “cs411\_507\_hw03\_yourname.zip”

1. (**20 pts**) Consider GF(28) used in AES with the irreducible polynomial p(x) = x8+x4+x3+x+1. You are expected to query the server “cryptlygos.pythonanywhere.com/poly/*<your\_id>*”, which will send you two binary polynomials a(x) and b(x) in GF(28). Polynomials are expressed as bit strings of their coefficients. For example, p(x) is expressed as '100011011'. You can use the Python code “**Q1\_student.py**” given in the assignment package to communicate with the server.
   1. (**10 pts**) You are expected to perform c(x) = a(x)×b(x) in GF(28) and return c(x) as bit string.

I used python BitVector module for this question. The value of c(x) is 01110011. After sending the value of c(x), I received Congrats message. The codes could be reached from the submitted Q1\_student.py file.

Calculation process:

n = 8

modulus = BitVector.BitVector(bitstring = ‘100011011’) # irreducible polynom

bit\_a = BitVector.BitVector(bitstring = a)

bit\_b = BitVector.BitVector(bitstring = b)

c = bit\_a.gf\_multiply\_modular(bit\_b,bitstring = b) # multiply bit\_a and bit\_b and reduce it with the irreducible polynom if necessary.

* 1. (**10 pts**) You are expected to compute the multiplicative inverse of a(x) in GF(28) and return a-1(x).

I used python BitVector module for this question. The value of the multiplicative inverse of a(x) is 10100000. After sending the value of the multiplicative inverse of the a(x), I received Congrats! message. The codes could be reached from the submitted Q1\_student.py file.

Calculation process:

a\_inv = bit\_a.gf\_MI(modulus,n) # take the multiplicative inverse of a

1. (**30 pts**) Consider the Geffe generator of three LFRSs (LFSR1, LFSR2, and LFSR3) with the following connection polynomials:

C1(x) = x14 + x5 + 1

C2(x) = x17 + x3 + 1

C3(x) = x11+ x2 + 1

You also observed the following output sequence of the Geffe generator:

z = [0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1]

Can you find the initial states of LFSR1, LFSR2, and LFSR3?

I used correlation attack to find the initial states of the LFSR1 and LFSR3. In the first step, I generated all possible initial states for LFSR1 and LFSR3, then for each initial state, I send these initial states and their corresponding connection polynomial values to the LFSR function. At each iteration of the LFSR method, I compared the returned value of the LFSR function with the corresponding value of the Geffe output. Then the initial state that gives me highest correlation ratio with the Geffe output would be initial state of the LFSR1 and LFSR3.The initial state of the LFSR2 is determined by the using the maximum correlation sequences of LFSR1, LFSR3 and the Geffe function. The initial state that results in the exact value of z when (x1 & x2) ^ (x2 & x3) ^ (x3) is performed would be the initial state of the LFSR2. The codes could be reached from the submitted Q2\_student.py file.

Initial state of LFSR1: [1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1]

Initial state of LFSR2: [0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0]

Initial state of LFSR3: [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]

1. (**20 pts**) Consider the combining function given in the following table, that is used to combine the outputs of three **maximum-length** LFSR sequences:

F(x1, x2, x3) = x1x2 ⊕x1x3 ⊕ x2x3 ⊕ x1x2x3.

* 1. (**5 pts**) The lengths of LFSRs are 79, 85, and 97, respectively. Compute the linear complexity and the period of the output sequence.

Since the length of the LFSR’s are relatively prime with each other:

Linear\_complexity = 79\*85 + 79\*97 + 85\*97 + 79\*85\*97 = 673.978

Period = (279 - 1)\*( 285 - 1) + (279 - 1)\*( 297 - 1) + (285 - 1)\*( 297 - 1) + (279 - 1)\*( 285 - 1) \*( 297- 1) = 3705346855594118253554271520278013051304639509300339553647525022102584882626562

* 1. (**15 pts**) Analyze the function F in terms of three criteria:
* Nonlinearity degree
* Balance
* Correlation

Is this a good combining function? Explain your answer.

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | X3 | F |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

* Nonlinearity degree of F is **3**.
* Since the function generates more ‘0’ than ‘1’, the function is **not balanced**.
* x1: The correlation ratio of x1 with the output is 5/8.
* x2: The correlation ratio of x2 with the output is 5/8.
* x3: The correlation ratio of x3 with the output is 5/8.

Since the correlation ratio of the x1,x2,x3 is greater than 1/2, the function is not a good combining function. Because the output of the F is correlated with the value of the variables. Therefore, the predictability chance is higher than 50%.

1. (**20 pts**) Consider a modified AES without ShiftRow and Mixcolumn layers, where the secret key length is 128-bit. Show that with moderate effort you can break it.

If Shift Row is omitted in the AES, then instead of attacking 128 bits, attacker can consider 128 bits as 4 independent 32 bits block and attack these blocks separately to find the secret key. On the other hand, if Mixcolumn is omitted, then the attacker would consider 128 bits secret key as 16 independent 8 bits block.

When there is not ShiftRow and Mixcolumn layers in the AES, one byte in the plain text will only affect the corresponding byte in the cipher text. For this reason, the encryption process will constituted of 16 substitution ciphers with the block size of 8.

In the case of chosen plain text attack, if we choose P0 = [0,0,0,….], P1 = [1,1,1,1,..,1], P2 = [2,2,2,..,2], P3 = [3,3,3….,3] and P255 = [255,255,255,..], then we can find the secret key and break the AES ciphers.

1. (**10 pts**) The cipher block chaining (CBC) mode has the property that it recovers from the errors (corruption, deletion, and insertion) in ciphertext blocks. Its encryption schemes are given as follows

Encryption primitive: Ci = EK(Pi ⊕ Ci-1)

Decryption primitive: Pi = DK(Ci) ⊕ Ci-1

How many blocks decrypt incorrectly if the ciphertext block Ci is corrupted during transmission? Show which plaintext blocks are corrupted.

At most 2 decrypt blocks will incorrectly decrypted. These are:

Pi = D(Ci) ^ Ci-1 and Pi+1 = D(Ci+1) ^ Ci

Since Ci is corrupted then D(Ci) may also be corrupted which affects the computation of Pi. In the second case, since we need the value of the Ci to compute Pi+1, when Ci is corrupted then Pi+1 may also be corrupted. The rest of the blocks will not get effected, since CBC mode has a self-synchronizing property.

**Exercise for Rainbow Tables (Non-credit question)**

Consider ten digests in the attached file “**rainbow\_table.py**”, each of which is the hash of a six-character password. Your mission is to find those passwords using the rainbow table given in the attached file “**rainbowtable.txt**”. Complete and submit the Python code in the file “**rainbow\_table.py**” such that it finds and prints out the ten passwords corresponding to the digests.