CS411 Cryptography Homework #2
Hakan Buğra Erentuğ 23637
Erkay Savaş 21 / 10 / 19

Before starting:

The all python codes that are attached, are my own work, unless I stated otherwise. You can run and check my results in your own computer to verify the solutions. I did not focus on comments because the codes that I designed is just helping me to find solutions. Because any code grading is not the case as far as I know. However, If you see anything suspicious, inform me and I can assure you I could explain what I did. You can find my python codes at the zip file. I also publish my codes on GitHub, to guarantee myself in case of any plagiarism. https://github.com/herentug/CS411

Solution #1.a

The key point of the question is primeness of 61 number. That fact makes us enable to calculate number of generators from fi(60). Mathematically, There have to be 16 different combination by 60*1/2*2/3*4/5 = 16. However, to illustrate visually, I placed my code output below. Please check numberGenerator.ipynb in zip file.

```
[2, 6, 7, 10, 17, 18, 26, 30, 31, 35, 43, 44, 51, 54, 55]
```

Solution #1.b

To find the solution we have to find, 5 different integer between the range (1,61) which can obtained from another numbers modulo expressions. To achieve that we can just search brutely because there are numbered possibility. Please check numberGenerator.ipynb to and verify the output. The output is 9, 20, 34 and 58. These numbers are the generators not the subgroup elements. Therefore logically, the 5^{th} element is 1. The subgroup output is [1,9,20,34,58].

Solution #2

The numbers p and q are prime, therefore we have to take the inverse modulo of phi(q,p). The gcd of c and n number is k = 1. Therefore, d number is modulo inverse (inverse modulo function is taken by internet) and power of c d in modulo n is given below. You are free the verify the outputs please check PhiCalculator.ipynb. The outputs are:

```
17780486146994356390536698036064157558306067628911641314354623865206367540771264269376489317932941129613874729594828696361809009003718595069818719707531861843667554232045010862980569632311263598996529228578729062647856589914115371678446914411535684158634995069502339626673711631508561804630291588176843797867
```

```
30256242323116471143377579036851734956599566784957808777900084159344311403348077243305468893900944356797865493742870723640786398026030923177601473087110374524033894222588248814856928431210421802802119552503278962630326720552941119001410709815232177730156585869309231777327139661729727635100740592709408204160
```

CS411 Cryptography Homework #2
Hakan Buğra Erentuğ 23637
Erkay Savaş 21 / 10 / 19

Solution #3

To decide whether has solution or not, we need to find a^-1*b=x mod n. In other saying if we have inverse of a we can find a solution. To check that property, we need do find exponential modulo and inverse of modulo which are very expensive operations. In my python notebook code, modinv and egcd functions have been taken from the internet. Please check the Question3Checker.ipynb for the rest of the code and verify the output. The outputs are given below.

There is no inverse of a for sub-question 1.
There is no inverse of a for sub-question 2.
The inverse of a for sub-question 3 is 87208981786492864261728145663
The solution for sub-question 3 is 327252728639173874206458501252

Question #4

In this question no other function has implemented except for lfsr.py which is given from the TA's. Therefore, I cannot address any python code to verify the solution. However, I can explain how the solution will be calculated. To find whether they generate maximum period sequence, first we have to check the functions primitiveness. Primitive function means that they cannot write down with two other polynomial multiplication. However, x^5 + x^2 + 1 is a primitive function, therefore no matter the initial state is it will always have same period sequence. Therefore we can just predict the initial state randomly and execute the lfsr.py to find the period. The output for period length value is 31, or it can be calculated by (2^(n))-1 which confirm the solution. However, for question 2, x^5+x^3+x^2+1 is not primitive because it can be write down as (x^3 + 1)*(x^2 + 1). The period length is depend on initial state. Therefore, two method can be applied, alpha substitution method which requires the handwork, or checking with various initial states and find the lcm. I changed the S the initial state in lfsr.py and find solution 1,2,3,4,6 and 12. Therefore, we can take the lcm of these number and calculate 12 as a maximum period length correctly and logically.

Question #5

To absolute unpredictability, the number of 1's and 0's have to be equal so anyone cannot say anything about balance. The all of the examples in Question 5, they have unbalanced number of bits. In example a and b there are 59-41. In example c the numbers 48-52. We can say, c is more unpredictable, however we cannot say neither of them, is not completely unpredictable.

Question #6

Please check the BonusQuestion.ipynb before the read instructors. The first method that assuming the C(x) length is 7 and trying all possible combinations and decrypt exhaustively. However, logically the possible keystream must be larger than 868 wichh is the length of ciphertext. Therefore the functions generateAllBinaryStrings and printTheArray is not

necessary. Also these functions is taken from geeksfromgeeks.com and never used for the rest of the solution. Firstly, we know the first 12 char of plaintext which is the very critical hint. First we used defined functions to obtain bitstreams of the string “Dear Student” as a hint. We can xor the hint and first 84 bits of ciphertext to find first 84 bit of key with
keyFirst84=xor(ctext[0:84],hint)
print(keyFirst84) and get the output (below).

[1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1]

However, we need to find the C(x) which is polynomial relation. To find that we can use BM function that already given. The output is the value L and C is given below

(17, [1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1])

Then we can calculate the key stream with S initial step which is first 17 bits of the ciphertext (in reverse, because of left shift logically) and we already know the C(x) with 18 bit length. Therefore we can call LFSR function that from you and find 868 length randomed key generator. The below output is 868 length keystream that needed to be XORing with our ciphertext.

Keystream = [1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0,
0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1,
0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1,
1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,
0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1,
0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1,
0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1,
1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,

CS411 Cryptography Homework #2
Hakan Buğra Erentuğ 23637
Erkay Savaş 21 / 10 / 19

```
1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,
1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1]
```

However we first check if the first 84 bit of key stream and bin2ASCII output is the same. print(keystream[0:84]==keyFirst84) that returns True. And the decrypted message is

Dear Student,
You have worked hard and that paid off:)
You have just earned 20 bonus points. Congrats!
Best,
Erkay Savas

Please check BonusQuestion.ipynb for more detail and please ignore all parts about exhaustive search which is a stupid attempt.