Q1 Team Name

0 Points

The_Kryptonians

Q2 Commands

5 Points

List the commands used in the game to reach the ciphertext.

go $\rightarrow$ go $\rightarrow$ go $\rightarrow$ go $\rightarrow$ go $\rightarrow$ give $\rightarrow$ read

Q3 Analysis

30 Points

Give a detailed description of the cryptanalysis used to figure out the password. (Use Latex wherever required. If your solution is not readable, you will lose marks. If necessary the file upload option in this question must be used TO SHARE IMAGES ONLY.)

After read command following hashed sequence appeared on the screen :
22 11 7 93 125 27 46 65 24 92 93 27 65 59 101 22 104 78 59 5 90 16 101 80 9 81 97 108 65 100 87 40
It is given password is made of letters between 'f' and 'u'. The letters are in alphabetic order. For hashing these letters is viewed as sequence of numbers $x_1, x_2 ... x_m$ in Field F_{127} . The i^{th} number of hashed sequence is power sum $x_1^{i-1} + x_2^{i-1} + ... + x_m^{i-1}$.

Observation

As for i=1 we get m = 22. So number of letters in the password are 22. We observed that as letters are in between 'f' to 'u' this implies that some of these letters are repeated in the password and therefore some of the numbers are repeated in the sequence.

Newton's identities

We used Newton's identities which give relations between two types of

symmetric polynomials, namely between power sums and elementary symmetric polynomials to solve for values of $x_1, x_2 ... x_m$.
Given $x_1^{i-1} + x_2^{i-1} + ... + x_n^{i-1}$ variables the k-th power sum is denoted by :
$p_k = \sum_{i=1}^{n} x_i^k$.
The elementary symmetric polynomials over Field F_{127} e_k using Newton's identities can be computed as :

$$ke_k = \sum_{i=1}^{k} (-1)^{i-1} e_{k-i}(x_1, x_2 ... x_n) p_i(x_1, x_2 ..... x_n)$$

Then the polynomial with roots $x_1, x_2 ... x_n$ can be expressed as :

P(x) = $\sum_{k=0}^{n} (-1)^k e_k x^{n-k}$

Approach

We ran a loop in which in each iteration we computed elementary symmetric polynomials using hashed sequence of unknown numbers and then formulated the polynomial. As x_i is in range of 0 to 126 , we check for each value between this range if it is a root of polynomial formed. After computing roots of the polynomial we calculated hashed sequence for remaining unknown numbers(repeated numbers) using values of already known numbers (As some numbers are repeated in the sequence so to find repeated numbers we eliminate already known numbers from the power sum equations)
In 4 iterations of the loop we found all 22 numbers of the sequence in the following way :

$1st\ iteration$	$103, 105, 107, 109, 110, 111, 112, 113, 114, 117$
$2nd\ iteration$	$105, 109, 110, 112, 113, 114$
$3rd\ iteration$	$105, 109, 112, 113$
$4th\ iteration$	$109, 112$

We then sorted the numbers in non-decreasing order as letters are in alphabetic order in the password.
We observed all the numbers are ASCII values of letters between 'f' to 'u'. After converting them to letters we got our password as :

Password : **giiikmmmmnnopppqqqrru**

References

1. https://en.wikipedia.org/wiki/Newton%27s_identities

No files uploaded

Q4 Password

15 Points

What was the final command used to clear this level?

giiikmmmmnnopppqqqrru

Q5 Codes

0 Points

It is MANDATORY that you upload the codes used in the cryptanalysis. If you fail to do so, you will be given 0 for the entire assignment.

Decrypt.py Download

```
1  def Coeff_e(p,n):
2      e=
   [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
3      for i in range(1,32):
4          sum=0
5          for j in range(1,i+1):
6              sum+= ((e[i-j]*p[j])%n *((-1)**(j-1)))
7              sum=sum%n
8          e[i]=sum%n
9          for s in range(n):
10             if (s*i)%n ==e[i]:
11                 e[i]=s
12                 break
13     return e
14
15 def Poly_roots(e,p):
16     roots=[]
17     for i in range(127):
18         sum=0
19         for j in range(p[0]+1):
20             sum += (i**(p[0]-j)%127*e[j])%127 *((-1)**j)
21             sum=sum%127
22         if(sum==0):
23             roots.append(i)
24             Total_roots.append(i)
25     return roots
26
27 def reduced_powersum(roots,p):
28     new_p=[p[0]-len(roots)]
```

```
    for i in range(1,32):
        sum=0
        for j in range(len(roots)):
            sum+= (roots[j]**i)%127
            sum=sum%127
        new_p.append((p[i]-sum)%127)
    return new_p


p = [ 22, 11, 7, 93, 125, 27, 46, 65, 24, 92, 93, 27, 65, 59, 101,
22 ,104 ,78, 59, 5, 90, 16, 101, 80, 9 ,81, 97, 108, 65, 100, 87,
40
]
m=p[0]
n = 127
Total_roots=[]
for i in range(32):
    e=Coeff_e(p,n)
    roots=Poly_roots(e,p)
    print(roots)
    p=reduced_powersum(roots,p)

    if len(Total_roots)==m:
        break

Total_roots.sort()
password=""
for i in range(len(Total_roots)):
    password+=chr(Total_roots[i])

print("Password is:",password)

```

Assignment 7

GRADED

GROUP

Pranshu Gaur
Maryam Raza Khan
Maulik Singhal
View or edit group

TOTAL POINTS

50 / 50 pts

QUESTION 1

Team Name — **0** / 0 pts

QUESTION 2

Commands — **5** / 5 pts

QUESTION 3

Analysis — **30** / 30 pts

QUESTION 4

Password — **15** / 15 pts

QUESTION 5

Codes — **0** / 0 pts