Q1 Team Name
0 Points

Noobs

Q2 Commands
10 Points

List the commands used in the game to reach the ciphertext

The commands used to reach the cipher text were as follows:
1. go
2. climb
3. pluck
4. c
5. back
6. give
7. back
8. back
9. thrnxxtzy
10. read
Note: Command number 4(c) is not a regular command. It was just asked by the prompt to enter c to continue, so, we mentioned it.

Q3 Analysis
50 Points

Give a detailed analysis of how you figured out the password? (Explain in less than 500 words)

We first gave 'go' and 'climb' commands to investigate the cave. We saw some mushrooms. We picked them using 'pluck' command. Then we were asked to press 'c' button to continue. We gave 'back' command to go back to the cave. Then we gave 'give' command to put the mushrooms in the hole. After this, we recieved a message from a poor spirit trapped in the hole. He said that say these magic words ''thrnxxtzy'' for the hidden door to become visible. Also, the door lies in the main chamber. So, we went to the main chamber by giving 'back' command two times. On reaching the main chamber now, we give the magic code: 'thrnxxtzy'. The hidden door appears with the glass panel next to it. We gave 'read' command to read the message there.
After doing so, we get the information that the password for this level is an element of the multiplicative group Z_p^* where p is a prime number. So, we studied about the multiplicative group and undertood it's basic concepts. We found three pairs of numbers of the form (a, password * g^a) where g was an element in Z_p^* and a was an integer. The g in each pair was the same. The pairs of numbers are given as follows:
(429, 4319555036182345198080087497 42)
(1973, 176325509039323911968355873643)
(7596, 984869714048619924872947226 13)
p=455470209427676832372575348833

Some basic information about multiplicative groups related to this question is described below:
In number theory, Z_p is the set of non-negative integers less than p {{0,1,2,3...p-1}}. Z_p^* is then a subset of this which is the multiplicative group for Z_p modulo p. The set Z_p^* is the set of integers between 1 and p that are relatively prime to p (means they do not share any common factors). If p is a prime number, then Z_p^* is the set contining values from 1 to (p – 1).
Eg. for number 12, the Z_p^* group will be {1, 5, 7, 11}.
If p is a prime number, then a special cycle property is observed. If we take a number y:
y=g^x mod p
If we select g from Z_p^* and where p is a prime number the result will be cyclic (where we keep repeating the output in a sequence). Eg. N=9, we can select 2,4,5,7 or 8, so let's select 7.
7^1 % 9= 7
7^2 % 9= 4
7^3 % 9=1
7^4 % 9= 7
O/p is {7,4,1,7,4,1,7,...}

Coming back to our question, we are given that p is prime, so, we'll have this cyclic property. Since, p is a prime number, every number less than p is element of group (Z_p^*). So, for all 'a', "password"g^a" is also element of Z_p.
We assigned variables to the given pair of values as follows:
(a1,r1)=(429,43195555036182345198080087497 42)
(a2,r2)=(1973,176325509039323911968355873643)
(a3,r3)=(7596,98486971404861992487294722613)

Then we wrote a code that generates inverse modulus(inv1,inv2,inv3) for values r1, r2 and r3. The Euclidean formula was basis of the algo for it. The Euclidean formula we used was:
x*p+y*r1=1
x*p+y*r2=1
x*p+y*r3=1
Here, y is the inverse of r1, r2 and r3.
inv1=70749996790223471732904681640
inv2=22894714947875260260635368 5125
inv3=10517174837159740961444519 4812

Now from the property of group, we have 3 equations:
r1=(password*g^a1)%p; ...Eqn 1
r2=(password*g^a2)%p; ...Eqn 2
r3=(password*g^a3)%p ...Eqn 3

Then we multiplied g^(a2+a3), g^(a1+a3), g^(a2+a1) to equation 1st 2nd and 3rd respectively. Then we applied inverse law to r1 ,r2 and r3.
After doing so, we got this equation:
r1*g^(a2+a3)=r2*g^(a1+a3)=r3*g^(a2+a1) (where * is operation of Z_p)
Solving it:
r3=r1*g^(a3-a1), r2=r1*g^(a2-a1), r3=r2*g^(a3-a2)
Substituting values:
r3=r1*g^(7167), r2=r1*g^(1544), r3=r2*g^(5623)
Then, we solve it further:
g^7167=r3*r1^(-1) ; g^5623=r3*r2^(-1) ; g^1544=r2*r1^(-1)
We took g^5623 and g^1544. Since gcd(5623,1544) is 1, the Extended Euclidean Algorithm can be applied to find the value of "g". According to the Extended Euclidean Algorithm, we can find x and y such that the below equation holds

ax+by=GCD(a,b)

Here
a=1544 and b=5623
and we have to find two numbers x and y such that
1544*x+5623*y=1 (since GCD(1544,5623)=1)

By applying the Extended Euclidean Algorithm, we found out that
x=−2298 and y=631

Raising both LHS and RHS to power of g, we get : g^(1544*x+5623*y)=g^1
Solving for LHS values:
(g^1544)^(-2298)=(111590994894663139264552154672)^(-2298) mod p
(g^1544)^(-2298)=6367334591911148292811 8052957

(g^5623)^631=(42041307425102202802727078 5553)^631 mod p
(g^5623)^631=3472670083898772983740176 7230

Substituting these values:
(g^1544)^(-2298)*(g^5623)^631=
(6367334591911148292811 8052957*3472670083898772983740176 7230) mod p

(g^1544)^(-2298)*(g^5623)^631=5256508541796331102769 4339

Thus, g^1=g=5256508541796331102769 4339 (using the Extended Euclidean Algorithm)

Now we can put the value of g in any one of the initial three equations to find out the password. We applied it in the first equation.

(password*(5256508541796331102769 4339)^429) mod p=
43195550361823451980800874 9742 mod p

password=(5256508541796331102769 4339)^(-429)*(43195550361823451980800874 9742 mod p)

password=((5256508541796331102769 4339)^(-429) mod p)*
((43195550361823451980800874 9742 mod p) mod p)

password=
(44295682031614869088930169661 5*43195550361823451980800874974 2)mod p

password=134721542097659029845273957

Then we have our password 134721542097659029845273957. Then we put the password as command and cleared this level.

Q4 Password
10 Points

What was the final command used to clear this level?

The final command used to clear this level is:
134721542097659029845273957

Q5 Codes
0 Points

Upload any code that you have used to solve this level

ModCrypt3.ipynb Download

```
In [1]:
def modInverse(a, m):
    m0 = m
    y = 0
    x = 1

    if (m == 1):
        return 0

    while (a > 1):

        # q is quotient
        q = a // m

        t = m

        # m is remainder now, process
        # same as Euclid's algo
        m = a % m
        a = t
        t = y

        # Update x and y
        y = x - q * y
        x = t

    # Make x positive
    if (x < 0):
        x = x + m0

    return x


# Driver code
a = 11226815350263531814963336315
m = 19807040628566084398385987581

# Function call
print("Modular multiplicative inverse is",
      modInverse(a, m))
```

```
Modular multiplicative inverse is 17983774594023309985368857902
```

```
In [ ]:
```

Assignment 3

GRADED

GROUP
Manu Shukla
Rishabh Lakhwani
Divyansh Bisht
View or edit group

TOTAL POINTS
70 / 70 pts

QUESTION 1	Team Name	**0** / 0 pts
QUESTION 2	Commands	**10** / 10 pts
QUESTION 3	Analysis	**50** / 50 pts
QUESTION 4	Password	**10** / 10 pts
QUESTION 5	Codes	**0** / 0 pts