Mehmet Yağız Çebişli - 28229
CS411 - HW4

Question-1

- It is stated that the query function returns m from its parameter c, when c is not equal to the given c from the get function. So instead of passing given c to the query function, we can simply pass c+N as a parameter for the query function, because if the query function returns plaintext, it can be also said that it returns c^d. So when we pass c+N it returns (c+N)^d which is equal to c^d over modulus N. In short, passing c+N also returns the plaintext.
- After getting the integer version of plaintext, I used to_bytes function and pass the result to the checker function and I got “Congrats” message.
-string version of plaintext = **Bravo! You found it. Your secret code is 2697**

Question-2

-I used online tool to factorize given N since it is not too large. Online tool: https://www.alpertron.com.ar/ECM.HTM

- q = 196826265417960486085322440316737141829
 p = 198204563364416691873648851641357924741
- Euler totient = (p-1)*(q-1), d = inverse of given e on modulus phi(N)
- After finding the we can decrypt with given function
- PIN: 5718

-The implementation can be found at “solution.py”.

Question-3

-Since we do not have the private key and the key space is too large we can not decrypt the ciphertext.

Question-4

-We can do a known plaintext attack since we have a plaintext-ciphertext pair.By computing inverse of message1 over modulus p we can obtain h^k via multiplying t1 with inv_m1. Then by obtaining inverse of h^k we can easily get message 2 via t1*inv_hk. The message is b'In sorrow, seek happiness.'

-The implementation can be found at “solution.py”.

Question-5

-For each entry in rainbow table, I started generating chain considering first element as a starting password and stopped generating more elements when I find the second element of the entry. As I find matching digests I saved the corresponding passwords until I find all digests.

-The implementation can be found at “rainbow_table.py”, it takes several minutes to find all digests.

-This is the output of my code:

```
1069336813336423737456764255447948362620798920731849654052135161755614920910
91 found digest -  9 -> YMTFTG
6531348280069912168979105656415958857232824310409970634681352827372880382179
9 found digest -  3 -> TJJYEA
1106973523056629093306063530920716384828722324460923371353787800924813203784
0 found digest -  7 -> FCVPPI
1634484223441496897315936728625368900034529467980640707053365865895477213238
6 found digest -  6 -> PKRJBA
2073345077851520626485201943794145151176912473811372451866141685012961931425
4 found digest -  8 -> ZQPAGD
8773359391572311991287612069572780862331103702058765431655114777404298967091
9 found digest -  5 -> VCWIZG
2648860899877611181282195523407805078338024070758437424036706814413927037856
6 found digest -  4 -> OTQKHJ
1104061294994486633148921026240480717511950870348333892806983858404050187972
45 found digest -  2 -> GFSECD
4623939272454030584377322346837100764978971400888872440457752296360652693566
3 found digest -  1 -> LSUDFG
6812948804201419511003831274263165656016940965713553204145828522341194894886
6 found digest -  0 -> OPXXZF
['OPXXZF', 'LSUDFG', 'GFSECD', 'TJJYEA', 'OTQKHJ', 'VCWIZG', 'PKRJBA', 'FCVPPI',
'ZQPAGD', 'YMTFTG']
```