Selin Eyupoglu
21090

CS411 HW 4

1.

a. Double encryption in RSA implies that we essentially raise m(message) to e1*e2=e3. Calculating e1*e2 and finding its inverse in mod(φ(n)) is not a harder problem than finding inverse of e1 and e2, and multiplying them thus double encryption does not increase security.

b. The results can be seen in the console output of question1.py

2.

We know that the plaintext has 4 digits since it is a PIN. Since there are 10000 possible PINs, brute force attack becomes somewhat feasible than trying to factor n which is too large. Starting with a seed (0000), I tried to calculate ciphertext of the possible PINs one by one. The code that I have written stops when the generated ciphertext is equal to the one given in the question. Then the PIN used to generate that ciphertext is the PIN we're looking for, which is 8128. The code I've written to solve this question can be found as question2.py

3.

a. It is given in the question that cp is congruent to kp^e mod n, thus we can write cp as cp = kp^e + m*n for some integer m. p divides kp^e and p|a*n (since n is p*q). We can say that p | cp and conclude that gcd(cp,n) = p since p divides both and only other factor of n is q. When cp and n is known, we can obtain p by calculating their gcd, i.e. we can factor n.

b. I found the gcd of n and cp which is equal to p, q is found by n//p. Run question3.py for p and q values.

Selin Eyupoglu
21090

4. a. If we choose k (similar to blinding in question 3)
such that $\gcd(k,N)=1$ (it has an inverse)
We know e, we encrypt $k \to k^e \pmod n$
$c' = c \cdot k^e \pmod n$

• we query the oracle for c' and get $m' = (c')^d \pmod n$

$m' = (c')^d \pmod n$

$m' \equiv (c\ k^e)^d \pmod n$

$m' = \underbrace{c^d}_{m} \cdot k \pmod n$

$m' \equiv m \cdot k \pmod n$

we know k, can calculate its inverse.

also known

from known

$m \equiv k^{-1} m' \pmod n$

4.

b. I chose k to be 31, I calculated its inverse in (modn) and generated the corresponding cipher then send it to oracle. I have pasted the oracle's output as oracleout then used it to find the actual message by multiplying by k's inverse in modn. Resulting integer can be seen as the console output of question4.py