Cryptography Assignment - 10
Pramod Aravind Byakod

Question 1:
Suppose we use the RSA cryptosystem with n, e from the last homework.
(a) If the cipher text c = your ID number (as a decimal integer), can you compute the plaintext?
(b) Suppose that you find the decryption key to be d = 879829162542850074748838973716462641470292321076843078870413133138541894315167534655428516005898396122103324293925057981802023330186106794090644952807381680714475934931163153 can you now compute the plaintext for your ID number?
(c) Can you factor n? If so, what are the prime factors? Use a program to explain your approach.
(d) Try 100 random a's. How many of them allow you to factor n? Estimate the probability of success.

With ID, 113436879, being a cipher text, it's not possible to compute the plaintext and is not safe.

With d value being such a huge number, we can compute the plaintext for ID number and is safe.

Program:

```
n=125953175678398351570149977764211035679420156938429586850000579961775054888014711050952194404928504160243324417202380
e = 65537
id = 113436879
R = Integers(n)
d= 879829162542850074748838973716462641470292321076843078870413133138541894315167534655428516005898396122103324293925505
flag = 0
for a in range(100):
    for i in range(100):
        k = (e*d-1)//(2^i)
        p = gcd(power_mod(a,k,n)-1,n)
        if(p!=1 and p!=n):
            print("a: "+str(a))
            print("p: "+str(p))
            print("q: "+str(n/p))
            flag = 1
            break
        if flag == 1:
            break
```

Output:
a: 7
p:
139583468203468547958790587305879573958759862470582467047605860298560237860276207823
q:
902350237456476285605762840521734742036702374527584276547628750176247692048048506720957432 7

Yeah, we can factor n. Above p and q values are the prime factors.

Program:

```
n= 125953175678398351570149977764211035679420156938429586850000579961775054888014711050952194404928504160243324417202380
d= 879829162542850074748838973716462641470292321076843078870413133138541894315167534655428516005898396122103324293925057
e = 65537
R = Integers(n)
suc_crt = 0
k = e*d-1
flag = 0
for i in range(100):
    if (k%2==0):
        k = k//2
    else:
        break
for i in range(100):
    a = randint(1,n);
    p = gcd(power_mod(a,k,n)-1,n)
    if(p!=1 and p!=n):
        suc_crt = suc_crt + 1
print("The probability of success is: "+str(suc_crt)+"%")
```

Output:
The probability of success is: 52%

Question 2:
Suppose that we decide to use= 65537 as the RSA public exponent. Can we use prime numbers that are congruent to 1(mod e) to generate n? Why? Find a prime satisfying:

- **p≡1(mode);**
- **21000≤p≤21004;**
- **The first 9 decimal digits of p is your ID number.**

Explain your approach.

We know that n=p*q, where p is the generator of n.
Considering p=1(mod e), in that case p-1 divides e. We can say, p-1=k*e(k>=1).
But according to the keys definition, we have gcd (e, (p-1)(q-1))=1. In our case, gcd (e, (p-1)(q-1)) = gcd (e, k*e*(q-1))= k*(q-1) !=1.

Therefore, gcd of e and (p-1)(q-1) is not 1, **we can't use this number to generate n**.

Program:

```
e = 65537
id = 113436879
P = 10^293*id
while (P < 2^1001):
    if(mod(P,e)==1):
        print P
        break
    else:
        P = P + 1
while (P < 2^1001):
    if(P.is_prime()):
        print 'The prime number p can be:\n'+str(P)
        P = P + e
    else:
        P = P + e
```

Output:
11343687900000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000030023
The prime number p can be:
11343687900000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000079759214393