D-dtpqpsu csoport

1. csoportmunka

A feladatokat az órán készített excel munkafüzetek segítségével prototipizáltuk, ami végül python környezetbe lett átültetve.

1. fázis: Az aláírt feladat visszafejtése.
Az általunk kapott számsorozatot egy numpy.array változóban tároltuk, ami lehetővé teszi, hogy egy lépésben minden tagon egyszerre hajtsunk végre műveleteket.

```
crptmsg=np.array([17605,10175,7212,1492,18115,1492,
```

```
e=11
n=20711
```

A lent látható decrypt függvény első lépésben önmagukkal szorozza össze a megadott sorozat tagjait és az eredmény n modulusát egy átmeneti változóban tárolja. Ezt követően e-1 lépésben ismétli meg a folyamatot, de a megadott sorozat tagjait mindig a legutolsó iterációban felülírt átmeneti változóban tárolt maradékok sorozatával szorozza össze, és veszi az n modulusát. Az e-1. lépésben kapott sorozat lesz a visszafejtés eredménye.

```
#Dekódolás a nyilvános kulccsal
def decrypt(series,n,e):
    ser0=series
    sertemp=series
    for nol in range(e-1):
        sertemp=np.mod(ser0*sertemp,n)
    return sertemp
```

```
dcrptmsg=decrypt(crptmsg,n,e)
```

Az üzenetet az excelben található KARAKTER() függvény python megfelelőjével alakítottuk vissza.

```
msg=''.join([chr(i) for i in dcrptmsg])
```

Az utolsó iterációban kapott számokat egyenként behelyettesítve és a kapott betűket egy string változóba összefűzve az alábbi szöveget kaptuk:

```
'AcsoportnakkészítsenekegyRSAkulcspárt,aholad1000-
10000közöttiérték,majdacsoportvezetõadjabeegyüttanyilvánoskul
csukatvalamintacsoportnevétisazénszámomratitkosítva-
csakénolvashassamel!Amegoldásmenetétmutassákbe.'
```

2. fázis: RSA kulcspár készítése (1000<d<10000)

```
r=67
q=83
n=r*q

r1=r-1
q1=q-1
fi=r1*q1

#e>1 egész szám, amely kisebb, mint ϕ(n) és relatív prím ϕ(n)=(p-1)(q-1)-hez.
e=7
```

Az általunk megadott r és q (q>r) prím számoknak megfelelő d értéket az alábbi függvény segítségével kerestük. A megadott fi és e értékek figyelembevételével egy átmeneti változóban tárolt fi+1 értéket minden egyes iterációban további fi értékkel növeli a függvény, ha annak e modulusa nem egyenlő nullával. Amennyiben a feltétel teljesül, az iteráció megszakad, és a függvény kiírja a bemeneti paraméterekhez tartozó d értéket. A túl hosszú vagy végtelen számú lefutások elkerülésének érdekében, ha a ciklusok száma eléri a kétezret, a függvény automatikus leáll, és az „Error” üzenetet írja ki.

```
def def_d(fi,e):
    ntemp=fi+1
    cntr=1
    while np.mod(ntemp,e)!=0 and cntr<2000:
        ntemp+=fi
        cntr+=1
        clear_output()
        print("{}:{}".format(cntr,ntemp))
    if cntr<2000:
        clear_output()
        print("{}:{}".format(cntr,ntemp))
        return ntemp/e
    else:
        clear_output()
        print("Error")
        return 0
```

```
d=int(def_d(fi,e))
print('d={}'.format(d))
```

```
6:32473
d=4639
```

```
print("Nyilvános kulcs:({},{})".format(e,n))
print("Titkos kulcs:({},{})".format(d,n))
```

```
Nyilvános kulcs:(7,5561)
Titkos kulcs:(4639,5561)
```

3. fázis: A csoport nevének és nyilvános kulcsának titkosítása a tanár úr számára

A kért adatokon karakterenként haladva az excel KÓD() függvény python megfelelőjével azt számsorozattá alakítottuk.

```
uzenet = "D-dtpqpsu:(7,5561)"
```

```
asciimsglst=pd.Series([ord(i) for i in uzenet])
```

A megadott nyilvános kulcsot felhasználva és az alábbi encrypt függvény segítségével titkosítottuk a kért adatokat. A „feltörésétől” eltekintve a számsor csak a privát kulcs ismeretében visszafejthető, csupán a tanár úr számára olvasható. A függvény a számsorrá alakított üzenet minden egyes tagját e hatványra emeli és azok n modulusát adja eredményül egy új sorozat formájában.

```
def encrypt(series,n,e):
    return np.mod(np.power(series,e),n)
```

```
e=11
n=20711
```

```
crptmsg=encrypt(asciimsglst,n,e)
```

Az így kapott számsor lesz az első csoportmunka eredménye, az alábbi titkosított üzenet:

```
print(crptmsg.values.tolist())
```

[18580, 15680, 168, 10861, 13018, 5111, 13018, 11835, 11535, 5635, 4995, 7431, 19922, 3159, 3159, 2850, 16515, 4598]