code

December 1, 2022

1 Lab-3

[245]:
```
import math
from decimal import Decimal, getcontext
```

2 Basic number theory algorithms

[246]:
```
def convert_to_binary_modify(number):

    binary_representation=[]

    while number>0:

        binary_representation.append(number & 1)
        number = number >> 1

    return binary_representation
```

[247]:
```
def mod(number,module):

        numb_by_module=0

        if (number < 0):

                if (number // module != 0):

                        numb_by_module = module - (abs(number) - module *␣
 ↪(abs(number) // module))

                else:

                        numb_by_module = module - abs(number)


        elif (number // module != 0):

                numb_by_module = number - module * (number // module)

        else:

                numb_by_module = number


        return numb_by_module
```

[248]:

```
def gcd_extention( a, b):
    q0=0
    r0=0
    r1=0
    r2=0
    u0=0
    u1=0
    u2=0

    if ((a == 0) and (b ==  0)):
        u = v = 0
        return 0,u,v

    r2 = -a if (a < 0) else a

    r1 = -b if(b <  0) else b

    u2 = 1

    u1 = 0

    while (r1 != 0):

        q0 = r2 // r1
        r0 = r2 - q0 * r1
        r2 = r1
        r1 = r0
        u0 = u2 - q0 * u1
        u2 = u1
        u1 = u0

    u = u2

    if (a <  0):
        u = -(u)

    v =  0 if(b ==  0) else (r2 -(u) * a)//b

    return r2,u,v
```

[249]:

```
def inverted_element(number,moduls):

        number = mod(number, moduls)

        gcd,u,v=gcd_extention(number, moduls)

        if (gcd ==1):

                b0 = moduls
                t=0
                q=0
                x0 = 0
                x1 = 1
                if (moduls == 1):
                        return 1
                while (number > 1):
                        q = number // moduls
                        t = moduls
                        moduls = number % moduls
                        number = t
                        t = x0
                        x0 = x1 - q * x0
                        x1 = t

                if (x1 < 0): x1 += b0

                return x1

        if (gcd > 1):

                return "Roots more or does`t exist!"
```

[250]:

```
def horner_method(number,degree,module):

    binary_representation=convert_to_binary_modify(degree)

    result = 1

    for i in range(len(binary_representation)-1,-1,-1):

        result = (result * result) % module

        if (binary_representation[i] == 1):
            result = (result * number) % module

    return result
```

[251]:

```
def chines_remainder_theorem(a,m):

    M = 1
    x = 0

    for i in range(0,len(m)):
        M *= m[i]


    for i in range(0,len(m)):

        n = M // m[i]
        u = inverted_element(n % m[i], m[i])

        x += n *u * a[i]


    x %= M

    if (x < 0):
        x += M

    general_module = M

    return x,general_module
```

[252]:

```
def nthroot (n, A, precision):
    getcontext().prec = precision

    n = Decimal(n)
    x_0 = A / n
    x_1 = 1
    while True:
        x_0, x_1 = x_1, (1 / n)*((n - 1)*x_0 + (A / (x_0 ** (n - 1))))
        if x_0 == x_1:
            return x_1
```

3 Brute force

[253]:

```
def brute_force(cipher,exponent,module):

    text=0

    cipher=int(cipher,base=16)
    module=int(module,base=16)

    while(True):

        if(horner_method(text,exponent,module)==cipher):
            break

        text+=1

    return text
```

4 Attack with small exponent applied on hines reminder theorem

[254]:

```
def attack_with_small_exponent(exponent,c_list,n_list):

    c_list=[int(c,base=16) for c in c_list]
    n_list=[int(n,base=16) for n in n_list]

    x,general_module = chines_remainder_theorem(c_list,n_list)

    return hex(int(nthroot (exponent, x, 10000)))
```

[255]:

```
def check_result_attack_with_small_exponent(messages, exponent,n_list):

    n_list=[int(n,base=16) for n in n_list]

    for n in n_list:
        print(hex(horner_method(messages,exponent,n)))
```

[256]:

```
e = 3
C1 = "0x3115f665a5c62cfaeb9f3f0d2dfcfe8cafb4f90a005e20ea48d9b41607ef7188"
N1 = "0xEEB25A696A48E3DAAB70EC4C4BEF7C5998A07E465C90BD37F331F5BAA80011F9"
C2 = "0x243f9d1059312b9daa01cae439cfdab7a4035364b04e5a993e43a68b79636b36"
N2 = "0xAB5F12B623D023289CB3CAE70F1849808CE0C31F9733AD6F4AC2A5564DA84F2B"
C3 = "0x20f6e6410982d39289cc4eacc04ea2ce8c853dece720f78e88963c5343c4659c"
N3 = "0x9CBDBC7A89BB945021E1924C12A78122C0E0A7E8647AF5EDF9C47A9F021A5305"
```

[257]:

```
attack_with_small_exponent(e,[C1,C2,C3],[N1,N2,N3])
```

[257]:

```
'0x1ffffffffffffffff00633b0b2351cfcaa22b6539734270284c1d497c7891'
```

[258]:

```
check_result_attack_with_small_exponent(int(attack_with_small_exponent(e,[C1,C2,C3],[N1,N2,N3]
 ↪e,[N1,N2,N3])
```

```
0x3115f665a5c62cfaeb9f3f0d2dfcfe8cafb4f90a005e20ea48d9b41607ef7188
0x243f9d1059312b9daa01cae439cfdab7a4035364b04e5a993e43a68b79636b36
0x20f6e6410982d39289cc4eacc04ea2ce8c853dece720f78e88963c5343c4659c
```

[259]:

```
e = 5
C1 =␣
 ↪"0x6ccdb4011f5db8ef3f29f142a650e8708081a9ceeaf9d353266224737d0ca36b1422b0a3e361b922a0f32ee5
N1 =␣
 ↪"0xC50813768871D20E4FC91709C56928DF050BF4324073021770782E986CBBC4CC43B1FCF73A9D1F53E612F1D4
C2 =␣
 ↪"0x432051a354b60a8dd0310fbab2977bf760b7d4798ec913472af399d44da2349ccd3403fada1a3054d4c0c4d8
N2 =␣
 ↪"0xC630C3A7E71523F4AB5421DB6B36B391839EDDF7EEEF2421ECA9C791D7A248F3A9AEDA9B8A71E64064E862AE
C3 =␣
 ↪"0x7504e178dc9f193844be5ab5598353a341e415081ab4b84646cec00793ad4ca2a2cc20b8420c52d5967bae13
N3 =␣
 ↪"0xB6C30E7E05A427FC564903718759D30F811CE6567E95DCBC166E296F28E5CB7BB5CFDE535C6F1E22215ECD46
C4 =␣
 ↪"0x51aecd5864868da0f27513887a3efd1ce135519a4044afc6306d3dc40fa404f3aa41916ee71b4b2cddd08fd9
N4 =␣
 ↪"0xB2ACD8D627C83C60FFE9F413727872735E40905361B1F0C820806C272D92CFDEF39D18D174EEDEA91B9B2CCD
C5 =␣
 ↪"0x3dc5192d8c63fdfd3e6c97d3ade9e3d40314ca3b01b2b05f6f5b825b668107fa8d113d237db9776e5560b642
N5 =␣
 ↪"0xA79394E5ED5219FA8D701B8F8EEA2628D811FD7B087C08B73F4497D9B19E2039B61B4AF3FADDC7B2A11DB481
```

[260]:

```
attack_with_small_exponent(e,[C1,C2,C3,C4,C5],[N1,N2,N3,N4,N5])
```

[260]:

```
'0x1ffffffffffffffff0061d1f637a844983fe9225ee597bc228c4f52a9eb482c86ccfc59e1d00b
16d177320f95ba69e780760e4a91bfc57b807e18e2469c225e975e04a932ebb504137923aae00a7e
36d81d54370e9aeb1edafe4b89d1f48d6d02572429a65df972b8754452b2319d0939bece01c11311
d5c785a4951d5abc'
```

[261]:

```
check_result_attack_with_small_exponent(int(attack_with_small_exponent(e,[C1,C2,C3,C4,C5],[N1,
 ↪e,[N1,N2,N3,N4,N5])
```

```
0x6ccdb4011f5db8ef3f29f142a650e8708081a9ceeaf9d353266224737d0ca36b1422b0a3e361b9
22a0f32ee501d263ee73396f672b5f03d299e154fc6f26df78d8f9829a1f58659d1e22ef9237d323
e280a08bbda490a3b9e97bf989f3187a82229b993841a648743e319f5c7904bf3f7932457b9f2648
e360eefbb6a85e6764
0x432051a354b60a8dd0310fbab2977bf760b7d4798ec913472af399d44da2349ccd3403fada1a30
54d4c0c4d87d77a2334af0c090e0a86b3c0d5e4e251af71682547181a924b32c082db8180a09a421
```

```
670660fe79f4a4563a1801f2c3f8f908b69a0f30be7cf91142ec98ca91797d2ad258447b0a09086f
c7f1014d08081f4c46
0x7504e178dc9f193844be5ab5598353a341e415081ab4b84646cec00793ad4ca2a2cc20b8420c52
d5967bae13dd6d249ccf6cc92ee5ecf49514bf3b974d6e65f616e97bc9bebc06ff4b2c7d9922424a
498bc0ffbad975cbbdd1fea3d782588966b7e9c96a2ed3176ec726f43ce3e4de8c2e651011bb89b9
b31b00f69ea51c08aa
0x51aecd5864868da0f27513887a3efd1ce135519a4044afc6306d3dc40fa404f3aa41916ee71b4b
2cddd08fd997f1da6a66a55280f23761cb19aef76ac3d15123b10f26b2f876e0914995990b4bd695
37f472d7f6e13c6b7099cd74111655443aaaa60510536d51ffb976c576f7716e29baecbcad72eeb1
73dc230656c279749a
0x3dc5192d8c63fdfd3e6c97d3ade9e3d40314ca3b01b2b05f6f5b825b668107fa8d113d237db977
6e5560b6421b17a881a417b7d9e74101bb52289b683fb5bc0b19f5300817ba9ccbd213eb57c1e5a1
69a7b180c3d5df3616d13f15618c0feca4509a6b86946e91d97ac2ed17f4d3ee8cf6b085034043fb
10689f49345585cccf
```

5 Attack-in-the-Middle

[267]:
```
def attack_in_the_middle(l,cipher,exponent,module):

    cipher=int(cipher,base=16)
    module=int(module,base=16)

    X=[[i,horner_method(i,exponent,module)] for i in range(1,pow(2,l//2))]
    C_s=[(cipher*inverted_element(x[1],module)) % module  for x in X]

    T=0
    S=0
    T_e=0
    C_S=0

    for i in range(0,len(C_s)):

        triger=0

        for j in range(0,len(X)):

            if(C_s[i]==X[j][1]):
                T_e=X[j][1]
                C_S=C_s[i]
                triger=1
                break

        if triger==1:
            break

    for i in X:
        if(i[1]==T_e):
            T=i[0]
            break

    for i in X:
        if((cipher*inverted_element(i[1],module))%module==C_S):
            S=i[0]
            break

    return (T*S)%module
```

[271]:

```
def check_result_attack_in_the_middle(messages,expenent,modules):
    modules=int(modules,base=16)
    return hex(horner_method(messages,expenent,modules))
```

[272]:

```
e = 65537
l=20
C =␣
  ↪"0x1c2d97c113dd3bf4a699efb30ef01d12fc7d6d815b5b71fa7cdff26791d444484a0dddfe2ea677c2ea8c0b04
N =␣
  ↪"0xA1BD21600C6EEE61B966343EFF6BA4D6C1F6F55A1C3440CF7C59DC31692C2CA4F2279F790FE6168B70B14D6F
```

[273]:

```
attack_in_the_middle(l,C,e,N)
```

[273]:

```
620535
```

[274]:

```
check_result_attack_in_the_middle(attack_in_the_middle(l,C,e,N),e,N)
```

[274]:

```
'0x1c2d97c113dd3bf4a699efb30ef01d12fc7d6d815b5b71fa7cdff26791d444484a0dddfe2ea67
7c2ea8c0b0490b4da8612dd57c6282d64e08fc720573b4e0350'
```

[275]:

```
brute_force(C,e,N)
```

[275]:

```
620535
```

[276]:

```
e = 65537
l=20
C =␣
  ↪"0x366cf825d6dc4cb7c629722701d8e64bc48a4dadadacc7965b5869dd6ecb3cd3fd47b387180b75fa86b2f35a
N =␣
  ↪"0xB6DBAF4C7C1405E3F82D73A3F4B12661207066620BAAB6394DAF8225475F3E4D69FF84F540F124460991DEA3
```

[277]:

```
attack_in_the_middle(l,C,e,N)
```

[277]:

```
967415
```

[278]:

```
check_result_attack_in_the_middle(attack_in_the_middle(l,C,e,N),e,N)
```

```
[278]: '0x366cf825d6dc4cb7c629722701d8e64bc48a4dadadacc7965b5869dd6ecb3cd3fd47b387180b7
       5fa86b2f35a90900a1b3ad5b5e4e82cd25891acbb2ab8562cca4e2d273ba289b47fe0e5db0cd3453
       3932af39af990a7c9e3771046e363a84d8dc1023ac11f5e054164eba29f2dd0179b0b692cba4d763
       bb9b03a438fab671fb57c2a61b60d9f39a9bf03c6ff3c9632055f162dbcd9489bd8ef508a7ec1c96
       1f60f86af7b36e67c4cbfde300de35a77e8f455616fbcf46d37236939772d1d62450a25af249a8d6
       6f1f4ed90090cc63b3ef1ace7850dd06dafc247ab869a26cb987e7ba6b2833e48407c0365f631880
       324e3e0f10f3db434e7c41ef4b99bbe5bcc'
```

```
[279]: brute_force(C,e,N)
```

```
[279]: 967415
```

6 Sources:

https://rosettacode.org/wiki/Nth_root#Python