Coursework 1 - Exercise 9

November 2, 2023

In NISEC we have developed a new encryption scheme to help Alice communicate with Bob. The scheme works as follows:

To generate her keys Alice does the following:

- She chooses a prime p, which is used as a modulus;
- She chooses a generator g of a suitable group $\mathbb{G}$;
- She chooses a random x such that $2 \leq x \leq p-2$;
- She outputs her public/private key pair (pk, sk) as $pk = g^x$ and $sk = x$;
- Finally, she publishes the public parameters (p, g, pk), while keeping sk private.

For Bob to send a message to Alice, he does the following:

- He picks a message $m \in \mathbb{N}$;
- He computes $c = pk \cdot g^m$;
- He sends $c = (c1, g)$ to Alice.

For Alice to decrypt c, she does the following:

- Alice recovers $sk = x$;
- Alice computes $c1 \cdot g - sk = g^x \cdot g^m \cdot g - x = g^m$.

Moreover, recall that an encryption is called:

- Linear ciphertext homomorphic if:

$$\text{Enc}(pk_i, x_i) \cdot ... \cdot \text{Enc}(pk_n, x_n) = \text{Enc}(pk_1 \cdot ... \cdot pk_n, x_1 + ... + x_n)$$

- Linear key homomorphic if:

For n public/private key pairs

$$(pk_1, sk_1), ..., (pk_n, sk_n),$$

we can construct a new public/private key pair as:

$$(pk_1 \cdot ... \cdot pk_n, sk_1 + ... + sk_n)$$

a) Provide at least two drawbacks of our new encryption scheme;

b) Prove that our scheme is both linear ciphertext and key homomorphic.

For the rest of the exercise, assume that all encrypted messages have small values (i.e. they are not of cryptographic size).

We now assume that n different users independently encrypt n different values using our new encryption scheme. That is, the first user ecnrypts x_1 using pk_1, the second user encrypts x_2 using pk_2 and so on and so forth. As a next step, all n users send their ciphertexts to a cloud service provider, who already possess the sum $sk_1 + ... + sk_n$, but not each distinct sk_i.

c) Explain how the CSP can compute $x_1 + ... + x_n$.

---

a)

1. The proposed scheme has a decryption method that relies on being able to solve the Discrete Logarithm Problem(DLP) which is an intractable problem. For this reason, the hability of decrypt the message requires for m to be small
2. The encryption for this scheme requires a more computationally expensive operation, as well as the decryption method.

b)

[2]:
```python
from utils import generate_gp, generate_keys, discrete_log

G, P = generate_gp(nbits=1024, num_processes=8)

class NISEC_scheme():
    def __init__(self, sk, pk):
        self.sk = sk
        self.pk = pk
    def enc(self, m):
        return (self.pk*pow(G, m, P))%P
    def dec(self, c):
        return discrete_log(15, G, c*pow(G, -self.sk, P)%P, P)

alice = NISEC_scheme(*generate_keys(G, P))
bob = NISEC_scheme(*generate_keys(G, P))

# Asserting Encryption and Decryption
m = 14
c = alice.enc(m)
m_ = alice.dec(c)
print("m = {}, m' = {}".format(m, m_))
```

```
m = 14, m' = 14
```

[8]:
```python
# Asserting Linear homomorphic
m1, m2 = 14, 15
c1 = alice.enc(m1)
c2 = bob.enc(m2)
m3 = alice.dec(c1) + bob.dec(c2)
```

```
c3 = (c1*c2)%P

eve = NISEC_scheme((alice.sk+bob.sk)%P, (alice.pk*bob.pk)%P)
c3_ = eve.enc((m1+m2)%P)
m3_ = eve.dec(c3_)
print(f"c3 = {str(c3)[:10]}..., c3' = {str(c3_)[:10]}...")
print(f"{m1} + {m2} = {m1+m2} = {m3} = {m3_}")
if(c3 == c3_ and m3 == m3_):
    print("Linear homomorphic property is satisfied")
```

```
c3 = 9703111329…, c3' = 9703111329…
14 + 15 = 29 = 29 = 29
Linear homomorphic property is satisfied
```

c) Similarly as the previous calculations, we just need to generalize for n users:

[5]:
```
import numpy as np
N = 10

users = [NISEC_scheme(*generate_keys(G, P)) for _ in range(N)]
m = [i for i in range(N)]
sum_m = sum(m)
print(f"sum(m) = {sum_m}")
c = [users[i].enc(m[i]) for i in range(N)]

CSP = NISEC_scheme(sum([users[i].sk for i in range(N)])%P, np.prod([users[i].pk␣
  ↪for i in range(N)])%P)
print(f"CSP sum: {CSP.dec(int(np.prod(c)%P))}")
```

```
sum(m) = 45
CSP sum: 45
```

As we can see the LCH property that characterizes this scheme allowed us to compute the sum:

$$\mathrm{Dec}\left(\sum_i^N sk_i, \prod_i^N c_i\right) = \mathrm{Dec}\left(\sum_i^N sk_i, \prod_i^N \mathrm{Enc}(pk_i, x_i)\right) = \mathrm{Dec}\left(\sum_i^N sk_i, \mathrm{Enc}\left(\prod_i^N pk_i, \sum_i^N x_i\right)\right)$$