TIPE ▷ Listings

I Cryptosystème d'ElGamal

Code 1 – elgamal.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created in 2021

@author: Stanislas MEZUREUX
Copyright (c) 2021 Stanislas MEZUREUX. All rights reserved.
"""

import random

NUMBITS = 1024

# q -> cyclic group order
# g -> cyclic group generator
# x -> prvate key
# (q, g, h) where h = g**x mod q -> private key

def gcd(a, b):
    if a < b:
        return gcd(b, a)
    elif a%b == 0:
        return b;
    else:
        return gcd(b, a%b)


# Miller-Rabin
# medium.com/@prudywsh/how-to-generate-big-prime-numbers-miller-rabin-49e6e6af32fb
def is_prime(n, k=128):
    if n == 2 or n == 3:
        return True
    if n <= 1 or n % 2 == 0:
        return False
    s = 0
    r = n - 1
    while r & 1 == 0:
        s += 1
        r //= 2
    for _ in range(k):
        a = random.randrange(2, n - 1)
        x = pow(a, r, n)
        if x != 1 and x != n - 1:
            j = 1
            while j < s and x != n - 1:
                x = pow(x, 2, n)
                if x == 1:
                    return False
                j += 1
            if x != n - 1:
                return False
    return True


def generate_prime_candidate(length):
    p = random.getrandbits(length)
    p |= (1 << length - 1) | 1
    return p


def generate_prime_number(length=1024):
    p = 4
    while not is_prime(p, 128):
        p = generate_prime_candidate(length)
    return p


def keygen():
    q = generate_prime_number(NUMBITS)
    g = random.randint(2, q)
    x = random.randint(2**(NUMBITS-1), q)
    return (x, {'q': q, 'g': g, 'h': pow(g, x, q)})


def encrypt(n, pk):
    q, g, h = pk['q'], pk['g'], pk['h']
    r = random.randint(2**(NUMBITS-1),q)
    return {'c1': pow(g, r, q), 'c2': n*pow(h, r, q)}


def decrypt(n, x, pk):
    return (n['c2']*pow(n['c1'], -x, pk['q']))%pk['q']


def multiply(n1, n2):
    n1c1, n1c2 = n1['c1'], n1['c2']
    n2c1, n2c2 = n2['c1'], n2['c2']
    return {'c1': n1c1*n2c1, 'c2': n1c2*n2c2}
```

```python
def is_equal(n1, n2, sk, pk):
    d = {'c1': n1['c1']*pow(n2['c1'], -1, pk['q']),
         'c2': n1['c2']*pow(n2['c2'], -1, pk['q'])}
    return decrypt(d, sk, pk) == 1
```

II Secret Santa

Code 2 – SecretSanta.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created in 2021

@author: Stanislas MEZUREUX
Copyright (c) 2021 Stanislas MEZUREUX. All rights reserved.
"""

import smtplib
from random import shuffle
import copy
import secrets
import time
import elgamal as eg
from math import factorial

AMOUNT = 10
NAME = 'MPSI1 227/228'
DATE = '03/01/2022'
GMAIL_ADDRESS = 'secret.santa.tipe@gmail.com'
GMAIL_PASSWORD = '**secret(santa:)**'


class TooMuchInTheTeam(Exception):
    pass


def nb_participants_check(L):
    for i, team in enumerate(L):
        M = [e for A in L[:i]+L[i+1:] for e in A]
        if M != [] and len(M) < len(L[i]):
            raise TooMuchInTheTeam(f"Too much participants in {team[0][3]}")


def csv_to_list(data):
    with open(data) as f:
        L = f.read().splitlines()
    L = L[1:]
    for i, e in enumerate(L):
        L[i] = [i] + e.split(',')
    return L


def group_by_team(L):
    teams = []
    for e in L:
        if e[3] not in teams:
            teams.append(e[3])
    nb_teams = len(teams)
    M = [[] for _ in range(nb_teams)]
    for i, team in enumerate(teams):
        for e in L:
            if e[3] == team:
                M[i].append(e[0])
    return M


def make_pairs(L, pk):
    nb_teams = len(L)
    if nb_teams == 1:
        M = L[0].copy()
        shuffle(M)
        length = len(M)
        R = [(0, 0)]*length
        for i in range(length):
            R[i] = (eg.encrypt(M[i]+1, pk),
                    eg.encrypt(M[(i+1) % length]+1, pk))
        with open('/draw_files/secret_santa_draw.py', 'w') as f:
            f.write(f'draw = {R}')
        return R
    R = []
    M = copy.deepcopy(L)
    shuffle(M)
    L_new = copy.deepcopy(M)
    for i, team in enumerate(L_new):
        for j, e in enumerate(L_new[i]):
            if len(M) == 1:
                M_next = M[0]
            else:
                M_next = (M[:i]+M[i+1:])[(j+1) % (len(M)-1)]
            M_next_len = len(M_next)
            k = secrets.randbelow(M_next_len)
            gift_to = M_next[k]
            R.append((eg.encrypt(e+1, pk),
                      eg.encrypt(gift_to+1, pk)))
            with open('draw_files/secret_santa_draw.py', 'w') as f:
                f.write(f'draw = {R}\ndraw_len = {len(R)}')
            del M_next[k]
            if M_next_len == 1:
                M = [e for e in M if e != []]
    return R


def send_email(L, data, sk, pk, display_team=True):
    from_addr = GMAIL_ADDRESS

    server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
    server.set_debuglevel(1)
    server.ehlo

    server.login(GMAIL_ADDRESS, GMAIL_PASSWORD)

    for e_encrypted in L:
        e = (eg.decrypt(e_encrypted[0], sk, pk)-1,
             eg.decrypt(e_encrypted[1], sk, pk)-1)
        to_addrs = data[e[0]][4]
        subject = f"Secret Santa - {NAME}"
        text = (
            f'Bonjour {data[e[0]][1]},\nCette année, tu es en charge du '
            f'cadeau de {data[e[1]][1]} {data[e[1]][2]}'
            f'{" ("+data[e[1]][3]+")" if display_team else ""}. Je te rappelle '
            f'que le budget est de {AMOUNT}€ et que la célébration aura lieu '
            f'le {DATE}.\nJoyeux Nöel à toi !'
        )

        message = f"Subject: {subject}\nFrom: {from_addr}\nTo: {to_addrs}\n\n"
        message = message + text
        server.sendmail(from_addr, to_addrs, message.encode("utf8"))

        time.sleep(0.1)

    server.quit()


def zero_knowledge_proof(sk, pk):
    import draw_files.secret_santa_draw as ssd
    if ssd.draw_len == 1:
        return True
    gift_from = ssd.draw[0][0]
    gift_to = ssd.draw[0][1]
    for i in range(1, ssd.draw_len):
        gift_from = eg.multiply(gift_from, ssd.draw[i][0])
        gift_to = eg.multiply(gift_to, ssd.draw[i][1])
    fact = factorial(ssd.draw_len)
    for j in range(1, ssd.draw_len):
        for i in range(1, j):
```

```python
            c_i = ssd.draw[i]
            c_j = ssd.draw[j]
            if (eg.is_equal(c_i[0], c_j[0], sk, pk) or
                eg.is_equal(c_i[1], c_j[1], sk, pk) or
                    eg.is_equal(c_j[0], c_j[1], sk, pk)):
                return False
    return eg.decrypt(gift_from, sk, pk) == fact and eg.decrypt(gift_to, sk, pk) == fact


def Secret_Santa(data):
    try:
        sk, pk = eg.keygen()
        info = csv_to_list(data)
        L = group_by_team(info)
        nb_teams = len(L)
        nb_participants_check(L)
        R = make_pairs(L, pk)
        print(f'secret key : {sk}')
        print(f'public key : {pk}')
        send_email(R, info, sk, pk, nb_teams != 1)
    except TooMuchInTheTeam as TeamError:
        print(TeamError)


def resend(sk, pk, data):
    try:
        import draw_files.secret_santa_draw as ssd
        info = csv_to_list(data)
        nb_teams = len(ssd.draw)
        send_email(ssd.draw, info, sk, pk, nb_teams != 1)
    except ModuleNotFoundError as Error:
        print(Error)
```

III Dénombrement des tirages

Code 3 – draw_counter.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created in 2022

@author: Stanislas MEZUREUX
Copyright (c) 2021 Stanislas MEZUREUX. All rights reserved.
"""

from itertools import permutations
```

```python


"""
Presumed formula : (nb_people_in_one_team !)^(nb_teams)
"""


def check_draw(D, L, p, n):
    for e in D:
        if e[1] not in L[((e[0]-1)//p + 1)%n]:
            return False
    return True


def gen_draws(p, n):
    M = [i for i in range(1, n*p+1)]
    P = list(permutations(M))
    R = [[]]*len(P)
    for i, e in enumerate(P):
        D = [()]*(n*p)
        for j in range(len(e)):
            D[j] = (j, e[j])
        R[i] = D
    return R


def count(p, n):
    L = [[p*i+j+1 for j in range(p)] for i in range(n)]
    R = gen_draws(p, n)
    res = 0
    for e in R:
        if check_draw(e, L, p, n):
            res += 1
    return res

# print(count(10, 3))
```