BCS TP2

Exercice 1 : Factorisation

Q1.a

La consigne nous indique que le modulo a été généré avec un seul nombre premier. Ainsi n = p*p et p = racine_carre(n).

```
p = sqrt(n1)
phi = (p-1)*p
```

Q1.b

On récupère la clé privée d en calculant l'inverse modulaire entre e et phi.

d = inverse_mod(e, phi)

Q1.c

On calcule l'exponentiation modulaire du chiffré avec d et n1.

flag = power_mod(c1, d, n1)

==========Challenge 1==========

There is a reason we use two primes and not just one!

Q2

On peut trouver une explication simple de l'implémentation de l'algorithme sur ce lien : https://www.geeksforgeeks.org/pollard-p-1-algorithm/

Cette première étape de l'algorithme cherche un nombre premier dans l'espace de recherche défini par la variable bound, et grâce au calcul d'exponentiation modulaire et du gcd.

```
    a = 2
    for i in range(2, bound):
        a = power_mod(a, i, n)
        d = gcd(a-1, n)
        if 1 < d < n:
            return d
```

Q3

Une fois un premier nombre premier trouvé par la première partie de l'algo, on vérifie que le second facteur de n est premier. Une fois les deux facteurs trouvés on peut calculer phi

puis l'inverse modulaire entre e et phi avant de réaliser l'exponentiation modulaire sur le chiffré avec la clé privé obtenue.

```
p, q = 0, 0
r = n2
while True:
    p = pollard(n2, B)
    r = r//p
    if is_prime(r):
        q = r
        break
phi = (p - 1) * (q - 1)
d = inverse_mod(e, phi)
flag = power_mod(c2, d, n2)
```

==========Challenge 2==========

In practice, we often choose safe-primes: p = 2*p' + 1, with p' prime. This ensure that p-1 has a big prime factor, and avoid such problems

Exercice 2 : Decryption oracle

Ce post nous explique comment réaliser une attaque par chiffré connu : https://crypto.stackexchange.com/questions/2323/how-does-a-chosen-plaintext-attack-on-rsa-work

En utilisant un clair connu très faible (= 2) on peut alors forger un chiffré connu à envoyer dans l'oracle.

c_a = power_mod(2, e, n)
c_b = c_a * ct

L'oracle nous répond avec un nouveau message que l'on peut déchiffrer avec notre message choisi en clair (= 2) et obtenir le flag.

c_bd = 14584149962863613361710226942786189639239045571635188869085191500233431726447583568895532151523030
flag = (c_bd // 2) % n

==========Challenge decryption oracle==========
Well done! The flag is S1d3Ch4nn3l4tt4ck