Trabalho Final de Cripto 2020.1

Nome: Ramon Oliveira de Azevedo

Questão 1. Temos que a fórmula apresentada é a mesma que define um número de Fermat. Ao rodarmos o programa apresentado em aula, que realiza o teste de Miller-Rabin, temos que Fn irá cair na condicional marcada:

```
#Função dada em aula que realiza o teste de Miller_Rabin
def miller_rabin(num, b):
    b %= nums
    if b <= 1:
        return False

    q = num - 1
    k = 0
    while q % 2 == 0:
        k += 1
        q //= 2
    r = pow(b, q, num)

    print("Num: ", num)

    if r == 1 or r == (num - 1):
        return False
    i = 0
    while True:
        r = pow(r, 2, num)
        print("R: ", r)
        i += 1
        if r == 1:
            return True
        if i == k:
            return True
        #Abaixo, a condicional citada na resposta
        if r == (num - 1):
            return False

for n in range (0, 10):
    num = pow(2, pow(2, n)) + 1
    b = 2
    print(miller_rabin(num, b))
```

Como vimos, o número irá corresponder ao formato de r == (num - 1). Quando r entrar no loop while, ele terá o valor de 2. Logo, os números seguintes que serão o resultado de pow(r, 2, num), serão potências de 2. Observando o print abaixo, podemos notar que todos os números apresentados com o valor de r representam uma sequência, onde todos os números resultantes de pow(r, 2, num) pertencem a ordem de 2^(2^n). Como os números de Fermat podem ser definidos através da Fórmula Fn = 2^(2^n) + 1, podemos concluir q eles serão pseudoprimos de Miller Rabin, pois serão iguais a (num - 1). Além disso, vale afirmar que, como provado por Euler, o número da forma Fn = 2^(2^n) + 1 quando N = 5, é composto. E, até hoje, não se descobriu nenhum número F(n) primo com n > 4.

Questão 3.

a) Fatorando N, temos que ele é igual a: 89 . 101

Logo, podemos concluir que Phi irá ter ser o resultado da multiplicação de (p - 1) * (q – 1). Portanto, Phi irá valer 88 . 100 = 8800.

Por definição, temos que o E irá ter o valor do menor primo que não divide Phi. Assim, temos que E = 3. Para encontramos o expoente secreto D correspondente, basta realizarmos o Algoritmo de Euclides Estendido para acharmos o inverso de E (mod Phi). Como mostrado no print abaixo, temos que o valor será igual a 5864.

```
>>> def euclides_estendido(p, q):
    dividendo, divisor = p, q
    x_ant, y_ant = 1, 0
    x_novo, y_novo = 0, 1
    while divisor != 0:
        quociente, resto = divmod (dividendo, divisor)
        x_ant, x_novo = x_novo, (x_ant - x_novo*quociente)
        y_ant, y_novo = y_novo, (y_ant - y_novo*quociente)
        dividendo, divisor = divisor, resto
    return x_ant

>>> e = 3
>>> phi = 88 * 100
>>> d = euclides_estendido(e, phi)
>>> print(d)
-2933
>>> #Como d é negativo, temos que adicionar phi ao resultado final
>>> d += phi
>>> print (d)
5867
>>>
```

Questão 3

b) Usando a tabela dada como referência, temos que a mensagem 12345 será codificada como 112113114115116. Logo:

```
>>> texto_codificado = 112113114115116
>>> b1 = 1121
>>> b2 = 1311
>>> b3 = 4115
>>> b4 = 116
>>> print(pow(b1, e, n))
1404
>>> print(pow(b2, e, n))
6557
>>> print(pow(b3, e, n))
806
>>> print(pow(b4, e, n))
5799
>>>
```

Questão 5.
Resposta: Isso ocorre porque um Expoente Público Pequeno pode comprometer a segurança do RSA, isto é, colocar a privacidade das mensagens em risco.

Temos que o mdc(E, Phi) tem que ser igual a 1. Então, quando trabalhamos com números primos grandes, temos que phi será par, e o mdc de 2 com um número par não será igual a 1. A única possibilidade do E poder ser 2 é quando o P e o Q forem dois, pois assim o Phi seria igual a:

$$(2 - 1)(2 - 1) = 1 . 1 => 1$$

E, como sabemos, o mdc (2, 1) será igual a 1. Só que dessa forma, com P e Q sendo igual a 2, eles serão primos muito pequenos e isso é algo que pode prejudicar seriamente a segurança do RSA, perdendo então, a sua eficácia.

Questão 7.
Resposta: O teorema Chinês dos Restos nos serve para acelerar o processo de descriptação. Para tanto, é de suma importância que os dados calculados mostrados no enunciado, sejam realizados. Por exemplo:

Sabendo os valores de P e Q, podemos facilmente descobrir o valor de N, pois N = P . Q

Logo, pelo Pequeno Teorema de Fermat, temos que P | y^(p-1)-1. Dessa forma:

$$y^{(p - 1)} = 1 \pmod p \text{ e } y^{(q - 1)} = 1 \pmod q$$

Com isso, podemos expandir essa ideia até obtermos um sistema de congruências, tal que: x = a (mod p) e x = b (mod q)

Para tal, pelo Teorema Chinês dos Restos, temos que:

$$x = (a . q . q') + (b . p . p') \pmod n$$

Onde p' é o inverso de p em módulo q e q' é o inverso de q módulo p. Esses, por sua vez, são dois dos quatro dados armazenados pelo usuário.

Tomando como base o programa apresentado como resolução na Questão 12, temos que as duas formas reduzidas são aplicadas no processo para descobrir a congruência na qual a operação é equivalente, sendo utilizadas como expoentes na hora de calcular a aritmética modular.

A vantagem de usar o TCR no processo de descriptação é a simplificação do processo, tornando-o mais ágil.

Questão 8.

Segue a decodificação:

2823	=	b
2688	=	i
398	=	c
4335	=	h
2273	=	o

Obs: Coloquei o programa que realiza o procedimento em anexo