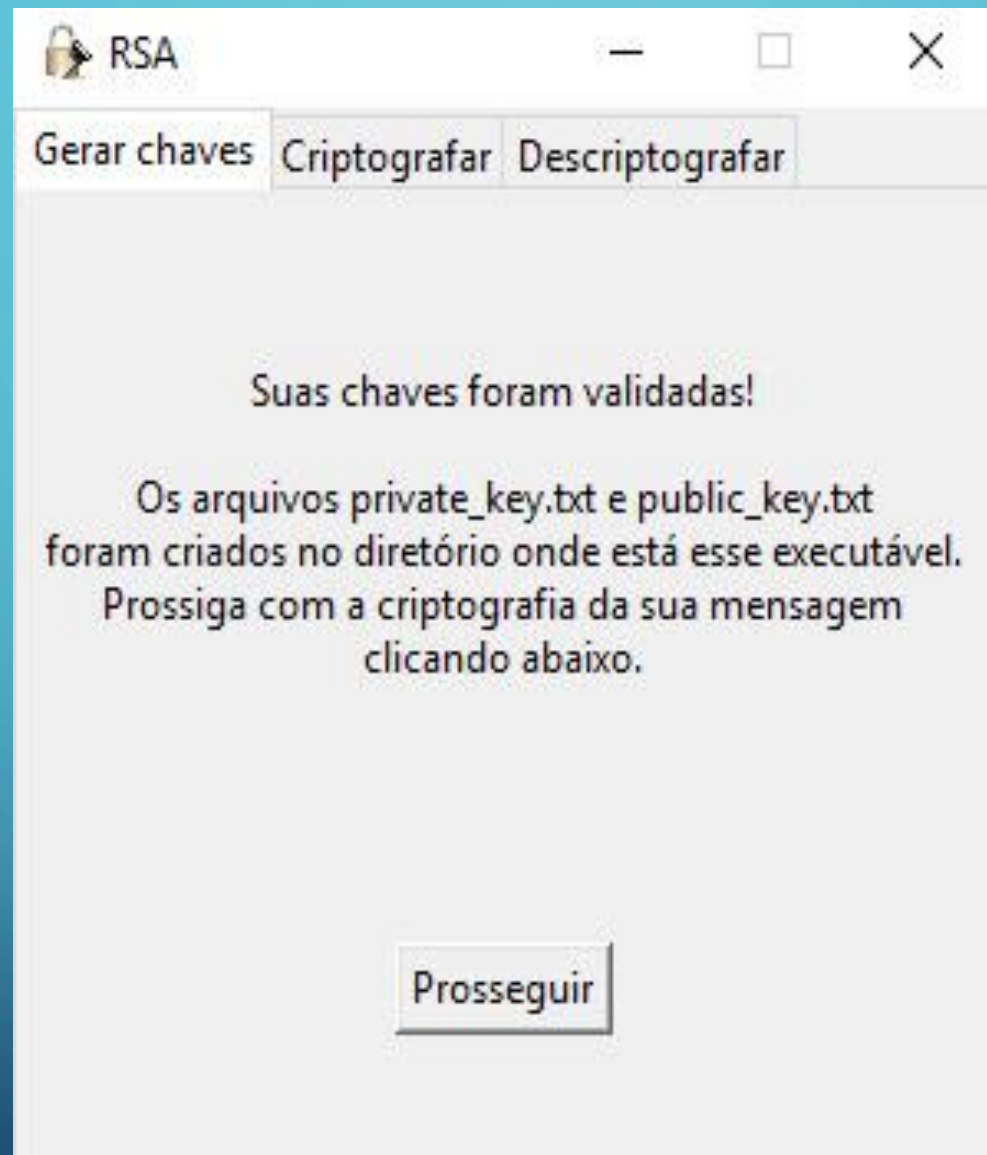





ALGORITMO DE CRIPTOGRAFIA/DESCRIPTOGRAFIA RSA

GUI





 RSA

Gerar chavesCriptografarDescriptografar

Digite os valores de P, Q e de um co-primo
a esses dois números, nessa ordem.
P e Q devem ser diferentes


Validar

RSA

Gerar chaves Criptografar Descriptografar

Insira 'N', 'E' e sua mensagem
para ser criptografada

Criptografar

 RSA

Gerar chavesCriptografarDescriptografar

Digite os valores de E, P e Q nessa ordem
para descriptografar seu arquivo

Descriptografar

GERAR PRIMOS

```
def gen_prime(): #gerando primo aleatorio

    n = secrets.randbits(32)

    while test_prime(n) == False:
        n = secrets.randbits(32)

    return n

def phi(p, q): #funcao totiente
    return (p-1)*(q-1)

def co_primos(x): #funcao que retorna um numero co-primo do numero passado
    y = gen_prime()

    while euclides(x, y) != 1:
        y = gen_prime()

    return y
```


VERIFICAR SE É PRIMO

```
def euclides(num1, num2): #algoritmo utilizado para saber o mdc
    resto = num1 % num2
    if resto == 0:
        return num2
    return euclides(num2, resto)

def test_prime(n): #funcao que checa se um numero e primo
    s = int(math.sqrt(n)) + 3
    if n % 2 == 0: #se o numero for par ja retorna que ele nao e primo
        return False
    for x in range(3, s, 2):
        if n % x == 0:
            return False
    return True
```


EXPONENCIAL MODULAR RÁPIDA

```
def exp_mod_rap(list_int_pow, d, mod, exp, r): #funcao de exponenciacao modular rapida
    if list_int_pow[-1] >= exp:
        if exp == 1:
            r = d % mod
            if exp in list_int_pow:
                return r * exp_mod_rap(list_int_pow, d, mod, (exp*2), r)

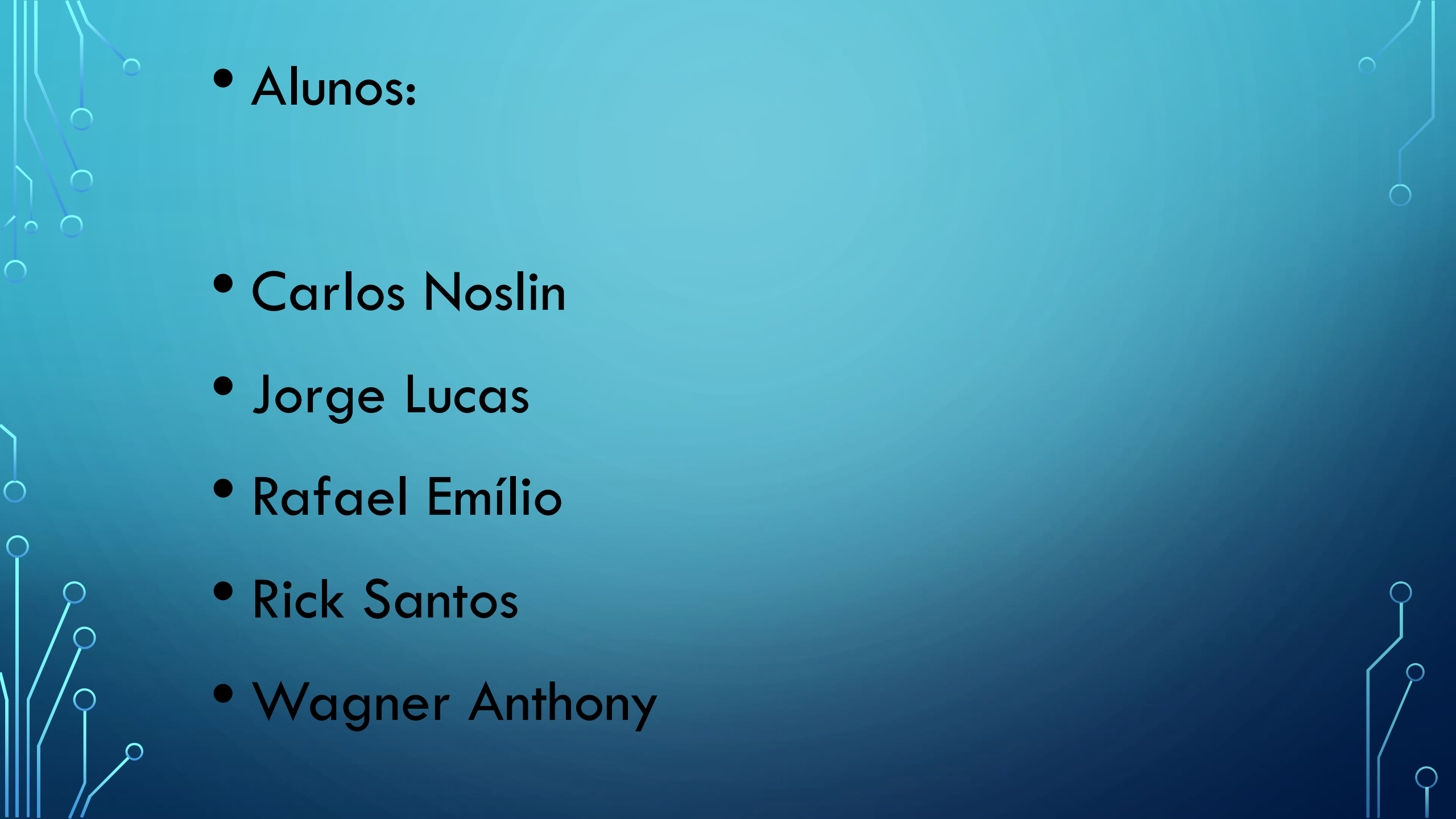
        else:
            r = (r*r) % mod
            if exp in list_int_pow:
                return r * exp_mod_rap( list_int_pow, d, mod, (exp*2), r)

            return exp_mod_rap(list_int_pow, d, mod, (exp*2), r)

    else:
        return 1
```

INVERSO MULTIPLICATIVO

```
def invert(a, b):  
    if a == 0:  
        return (b, 0, 1)  
    else:  
        g, y, x = invert(b % a, a)  
        return (g, x - (b // a) * y, y)  
  
def inversom_m(a, m): #funcao que retorna o inverso modular, ou inverso mutiplicativo  
    g, x, y = invert(a, m)  
    if g != 1:  
        raise Exception('Nao existe inverso modular')  
    else:  
        return x % m
```

- 
- Alunos:
 - Carlos Noslin
 - Jorge Lucas
 - Rafael Emílio
 - Rick Santos
 - Wagner Anthony