Guía de funciones RSA: Explicaciones y fundamentos matemáticos

Proyecto `rsa_basic`

Introducción

Este documento recoge las funciones desarrolladas en el proyecto `rsa_basic`, junto con una explicación detallada de su funcionamiento y los fundamentos matemáticos que las respaldan. Se trata de una guía que actúa como memoria de prácticas y servirá de referencia para consultas futuras.

Funciones desarrolladas

1. `gcd(a: int, b: int) ->int`

Descripción: Calcula el máximo común divisor (MCD) entre dos números enteros positivos usando el algoritmo de Euclides.

Implementación:

```
def gcd(a: int, b: int) -> int:
    while b != 0:
        a, b = b, a % b
    return a
```

Fundamento matemático:

Dado dos números enteros positivos a y b, el algoritmo de Euclides se basa en la propiedad:

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

Iterando esta propiedad hasta que el segundo operando sea 0, se obtiene el MCD.

2. `modinv(a: int, m: int) ->int`

Descripción: Calcula el inverso modular de a módulo m, es decir, un entero x tal que:

$$(a \cdot x) \bmod m = 1$$

Implementación:

```
def modinv(a: int, m: int) -> int:
    m0, x0, x1 = m, 0, 1
    while a > 1:
        q = a // m
        a, m = m, a % m
```

```
        x0, x1 = x1 - q * x0, x0
    if a != 1:
        raise ValueError("No existe inverso modular para los valores 
            elegidos")
    return x1 % m0
```

Fundamento matemático:
El inverso modular existe si y solo si $\gcd(a, m) = 1$. Se utiliza el algoritmo extendido de Euclides, que encuentra coeficientes x e y tales que:

$$ax + my = \gcd(a, m)$$

Si $\gcd(a, m) = 1$, entonces $ax \equiv 1 \mod m$, y por tanto x es el inverso modular buscado.

3. generate_keys(bits: int = 16)

Descripción: Genera un par de claves RSA (pública y privada) a partir de dos números primos de `bits` bits.
Implementación:

```
def generate_keys(bits: int = 16):
    p = getPrime(bits)
    q = getPrime(bits)
    while q == p:
        q = getPrime(bits)
    n = p * q
    phi = (p - 1) * (q - 1)
    e = random.randrange(2, phi)
    while gcd(e, phi) != 1:
        e = random.randrange(2, phi)
    d = modinv(e, phi)
    return (e, n), (d, n)
```

Fundamento matemático:
El sistema RSA se basa en:

- La dificultad de factorizar un número grande $n = pq$
- La existencia del inverso modular d de e módulo $\varphi(n) = (p - 1)(q - 1)$

La clave pública es (e, n), y la privada (d, n), donde:

$$d \equiv e^{-1} \mod \varphi(n)$$

4. encrypt_message(message: str, public_key: tuple) ->list[int]

Descripción: Cifra un mensaje utilizando la clave pública RSA. Transforma cada carácter en su código ASCII y aplica la fórmula de cifrado.
Implementación:

```
def encrypt_message(message: str, public_key: tuple) -> list[int]:
    e, n = public_key
    encrypted = []
```

```
    for char in message:
        m = ord(char)
        c = pow(m, e, n)
        encrypted.append(c)
    return encrypted
```

Fundamento matemático:
Cada carácter del mensaje se convierte en un entero m, y se cifra como:

$$c = m^e \mod n$$

El resultado es una lista de enteros cifrados.

5. decrypt_message(ciphertext: list[int], private_key: tuple) ->str

Descripción: Descifra una lista de enteros cifrados usando la clave privada RSA.
Implementación:

```
def decrypt_message(ciphertext: list[int], private_key: tuple) -> str
    ↪ :
    d, n = private_key
    decrypted = ""
    for c in ciphertext:
        if c >= n:
            raise ValueError(f"El␣valor␣cifrado␣{c}␣no␣puede␣
                ↪ descifrarse:␣debe␣ser␣menor␣que␣n={n}")
        m = pow(c, d, n)
        decrypted += chr(m)
    return decrypted
```

Fundamento matemático:
El descifrado aplica la función inversa:

$$m = c^d \mod n$$

Si $c = m^e \mod n$, entonces:

$$c^d \equiv (m^e)^d \equiv m^{ed} \mod n \equiv m \mod n$$

por la propiedad $ed \equiv 1 \mod \varphi(n)$.
Cada entero descifrado m se convierte nuevamente a su carácter con `chr(m)`.

Observaciones

- Este documento actúa como referencia viva. Se podrá ampliar con nuevas funciones, ejemplos de uso, casos de prueba y referencias bibliográficas.
- Se incluirá también información teórica clave sobre criptografía, RSA y seguridad de clave pública en futuras versiones.

Fin de la Fase 1
