```
import secrets
import math
from sympy import isprime, mod_inverse
from math import gcd

# ---------- PRIME UTILITIES ----------

def generate_large_prime(bits):
"""Generate a random prime number with the given bit length."""
while True:
candidate = secrets.randbits(bits)
candidate |= 1 # force odd
candidate |= (1 << (bits - 1)) # force correct bit length
if isprime(candidate):
return candidate

# ---------- RSA FUNCTIONS ----------

def power_mod(base, exponent, modulus):
result = 1
b = base % modulus
while exponent > 0:
if exponent & 1:
result = (result * b) % modulus
exponent >>= 1
b = (b * b) % modulus
return result

def choose_e(phi):
e = 65537
if gcd(e, phi) == 1:
return e
e = 3
while gcd(e, phi) != 1:
e += 2
return e

def string_to_int(s):
return int.from_bytes(s.encode(), byteorder='little')

def int_to_string(i):
length = math.ceil(i.bit_length() / 8)
return i.to_bytes(length, byteorder='little').decode()

# ---------- RSA SETUP ----------
```

```
bits = 2000 # your required RSA size

print("Generating 2000-bit primes... this may take a moment.")
p = generate_large_prime(bits)
q = generate_large_prime(bits)

n = p * q
phi = (p - 1) * (q - 1)
e = choose_e(phi)
d = mod_inverse(e, phi)

print("Keys generated successfully.\n")

# ---------- ENCRYPTION ----------

message_str = input("Enter a message: ")
message = string_to_int(message_str)

if message >= n:
raise ValueError("Message too large for RSA modulus. Use a shorter message.")

print("\nPublic key component (e):", e)
entered_e = int(input("Enter the public key component: "))

if entered_e != e:
raise ValueError("Incorrect public key component.")

cipher = power_mod(message, e, n)

# ---------- DECRYPTION ----------

print("\nPrivate key component (d):", d)
entered_d = int(input("Enter the private key component: "))

if entered_d != d:
raise ValueError("Incorrect private key component.")

decrypted_int = power_mod(cipher, d, n)
decrypted = int_to_string(decrypted_int)

# ---------- OUTPUT ----------

print("\nCiphertext:", cipher)
print("Decrypted message:", decrypted)
```