PRACTICE EXERCISE 02:
RSA DECRYPTION WITH CHINESE REMAINDER THEOREM (CRT)

1. Programming Exercise: RSA Decryption with CRT

RSA decryption involves computing

$$x = y^d \bmod n, \quad \text{where } n = pq.$$

When n is large (e.g., 2048 bits), this computation is expensive. To accelerate RSA decryption and signature generation, we use the Chinese Remainder Theorem (CRT). Instead of performing one large modular exponentiation modulo $n = pq$, we perform two smaller exponentiations modulo p and q, followed by recombination.

1.1. **Task:** Implement RSA decryption using the Chinese Remainder Theorem

1.2. **Required Functions:**

(1) `rsa_decrypt(y, d, n)`: Standard RSA decryption
(2) `rsa_decrypt_crt(y, d, p, q)`: RSA decryption using CRT

1.3. **Test Data:**

$$\begin{aligned}
p &= 12345678901234567890123456869,\\
q &= 98765432109876543210987654323,\\
n &= p \cdot q,\\
e &= 65537,\\
d &= 183037555140763297287823421841341095154128759392745892977,\\
y &= 12345678901234567890.
\end{aligned}$$

1.4. **Deliverables:** Students must submit the following items:

- Source Code: Implement all required functions in appropriately structured `.py` files.
- README Report: Provide a concise report that includes:
  - Test results using both small and large prime numbers.
  - A comparison of execution times between the standard RSA implementation and the CRT-optimized version.
  - A list of all group members (names and student IDs).

The next sections provide details on the CRT method and a worked example.

2. RSA Decryption with CRT

To decrypt a ciphertext y using CRT, we follow three steps: Transformation to the CRT domain, Exponentiation in the CRT domain, and Inverse Transformation (Recombination).

Step 1: Transformation to the CRT Domain. We split the ciphertext y into two smaller residues:

$$\begin{aligned}
y_p &\equiv y \pmod{p}\\
y_q &\equiv y \pmod{q}
\end{aligned}$$

These represent the ciphertext in the CRT domain.

Step 2: Exponentiation in the CRT Domain. Reduce the private exponent:

$$d_p \equiv d \pmod{p-1},$$
$$d_q \equiv d \pmod{q-1}$$

Then compute two modular exponentiations:

$$x_p \equiv y_p^{d_p} \pmod{p},$$
$$x_q \equiv y_q^{d_q} \pmod{q}$$

Each computation is about half as large as the original one.

Step 3: Inverse Transformation (Recombination). Recombine the results using CRT:

$$x \equiv (q \cdot c_p) \cdot x_p + (p \cdot c_q) \cdot x_q \pmod{n}$$

where:

$$c_p \equiv q^{-1} \pmod{p},$$
$$c_q \equiv p^{-1} \pmod{q}$$

These coefficients can be precomputed.

3. Worked Example: RSA Decryption with CRT

Let the RSA parameters be given by:

$$p = 11, \qquad q = 13, \qquad n = pq = 143, \qquad e = 7, \qquad d = 103.$$

We will decrypt the ciphertext $y = 15$ using the CRT.

Step 1: Compute y_p and y_q.

$$y_p = 15 \bmod 11 = 4, \quad y_q = 15 \bmod 13 = 2$$

Step 2: Compute d_p, d_q, x_p, and x_q.

$$d_p \equiv 103 \bmod 10 = 3, \quad d_q \equiv 103 \bmod 12 = 7$$
$$x_p \equiv 4^3 \bmod 11 = 9, \quad x_q \equiv 2^7 \bmod 13 = 11$$

Step 3: Recombine to get x.

$$c_p = q^{-1} \bmod p = 13^{-1} \bmod 11 = 6$$
$$c_q = p^{-1} \bmod q = 11^{-1} \bmod 13 = 6$$

Then

$$\begin{aligned} x &\equiv (13 \cdot 6) \cdot 9 + (11 \cdot 6) \cdot 11 \pmod{143} \\ &\equiv 78 \cdot 9 + 66 \cdot 11 \pmod{143} \\ &\equiv 1428 \pmod{143} \\ &\equiv 141 \end{aligned}$$

The result of the decryption is $x = 141 = 15^{103} \pmod{143}$.