Course Name:	Applied Cryptography	Semester:	V
Date of Performance:	04 / 08 / 2025	DIV/ Batch No:	AC2
Student Name:	Aaryan Sharma	Roll No:	16010123012

Experiment No:5

Title: Understanding Asymmetric Key Cryptography Algorithms

Aim and Objective of the Experiment:

Implementation of RSA algorithm and understanding RSA cryptanalysis

COs to be achieved:

CO2: Demonstrate and implement various Cryptographic Algorithms for securing systems.

Books/ Journals/ Websites referred:

1. Stallings, W., Cryptography and Network Security: Principles and Practice, Second edition, Person Education
2. Forouzan, B. A. (2018). Cryptography and Network Security. McGraw-Hill Education.

Theory: Explain the following.

Explain the requirement of asymmetric key cryptography:

Asymmetric key cryptography is needed because it solves the major limitations of symmetric key cryptography, especially in secure key exchange and scalability. In symmetric cryptography, both sender and receiver must share the same secret key, which creates challenges in securely transmitting that key over insecure channels and managing many keys in large networks. Asymmetric cryptography uses a public key (shared openly) and a private key (kept secret) to enable secure communication without the need to exchange a secret key beforehand. This ensures:

1. Secure key distribution over insecure channels.
2. Confidentiality (only the private key can decrypt data encrypted with the public key).
3. Authentication & digital signatures (verifying the sender's identity).
4. Scalability in large systems, as each user only needs one key pair instead of separate keys for each communication partner.

RSA(Rivest-Shamir-Adleman) Algorithm is an asymmetric or public-key cryptography algorithm which means it works on two different keys: Public Key and Private Key. The Public Key is used for encryption and is known to everyone, while the Private Key is used for decryption and must be kept secret by the receiver. RSA Algorithm is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published the algorithm in 1977.

RSA Algorithm is based on factorization of large number and modular arithmetic for encrypting and decrypting data. It consists of three main stages:

1. Key Generation: Creating Public and Private Keys
2. Encryption: Sender encrypts the data using Public Key to get cipher text.
3. Decryption: Decrypting the cipher text using Private Key to get the original data.

1. Key Generation

- Choose two large prime numbers, say p and q. These prime numbers should be kept secret.
- Calculate the product of primes, n = p * q. This product is part of the public as well as the private key.
- Calculate Euler's Totien Function
  $\Phi(n)$ as $\Phi(n) = \Phi(p * q) = \Phi(p) * \Phi(q) = (p - 1) * (q - 1)$.
- Choose encryption exponent e, such that
  - $1 < e < \Phi(n)$, and
  - $\gcd(e, \Phi(n)) = 1$, that is e should be co-prime with $\Phi(n)$.
- Calculate decryption exponent d, such that
  - $(d * e) \equiv 1 \bmod \Phi(n)$, d is modular multiplicative inverse of e mod $\Phi(n)$. Some common methods to calculate multiplicative inverse are the extended Euclidean Algorithm, Fermat's Little Theorem, etc.

- We can have multiple values of d satisfying (d * e) ≡ 1 mod Φ(n) but it does not matter which value we choose as all of them are valid keys and will result in same message on decryption.
- Finally, the Public Key = (n, e) and the Private Key = (n, d).

2. Encryption: To encrypt a message M, it is first converted to numerical representation using ASCII and other encoding schemes. Now, use the public key (n, e) to encrypt the message and get the cipher text using the formula:

$C = M^e \mod n$, where C is the Cipher text and e and n are parts of public key.

3. Decryption

To decrypt the cipher text C, use the private key (n, d) and get the original data using the formula:
$M = C^d \mod n$, where M is the message and d and n are parts of private key.

RSA implementation Code and Output :

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

def mod_inverse(e, phi):
    def egcd(a, b):
        if a == 0:
            return (b, 0, 1)
        else:
            g, y, x = egcd(b % a, a)
            return (g, x - (b // a) * y, y)

    g, x, y = egcd(e, phi)
    if g != 1:
        raise Exception('Modular inverse does not exist')
    else:
```

```
        return x % phi

def generate_keys(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)

    e = 17
    if gcd(e, phi) != 1:
        raise Exception("e and phi(n) are not coprime!")

    d = mod_inverse(e, phi)
    return (n, e, d)

def encryption(message, e, n):
    return pow(message, e, n)

def decryption(ciphertext, d, n):
    return pow(ciphertext, d, n)

p = 61
q = 53
message = 23

n, e, d = generate_keys(p, q)
ciphertext = encryption(message, e, n)
decrypted = decryption(ciphertext, d, n)

print(f'Public Key (n, e): ({n}, {e})')
print(f'Private Key (n, d): ({n}, {d})')
print(f'Original Message: {message}')
print(f'Ciphertext: {ciphertext}')
print(f'Decrypted Message: {decrypted}')
```

```
PS D:\KJSCE\BTech\TY\Sem V\AC> python
Public Key (n, e): (3233, 17)
Private Key (n, d): (3233, 2753)
Original Message: 23
Ciphertext: 2037
Decrypted Message: 23
```

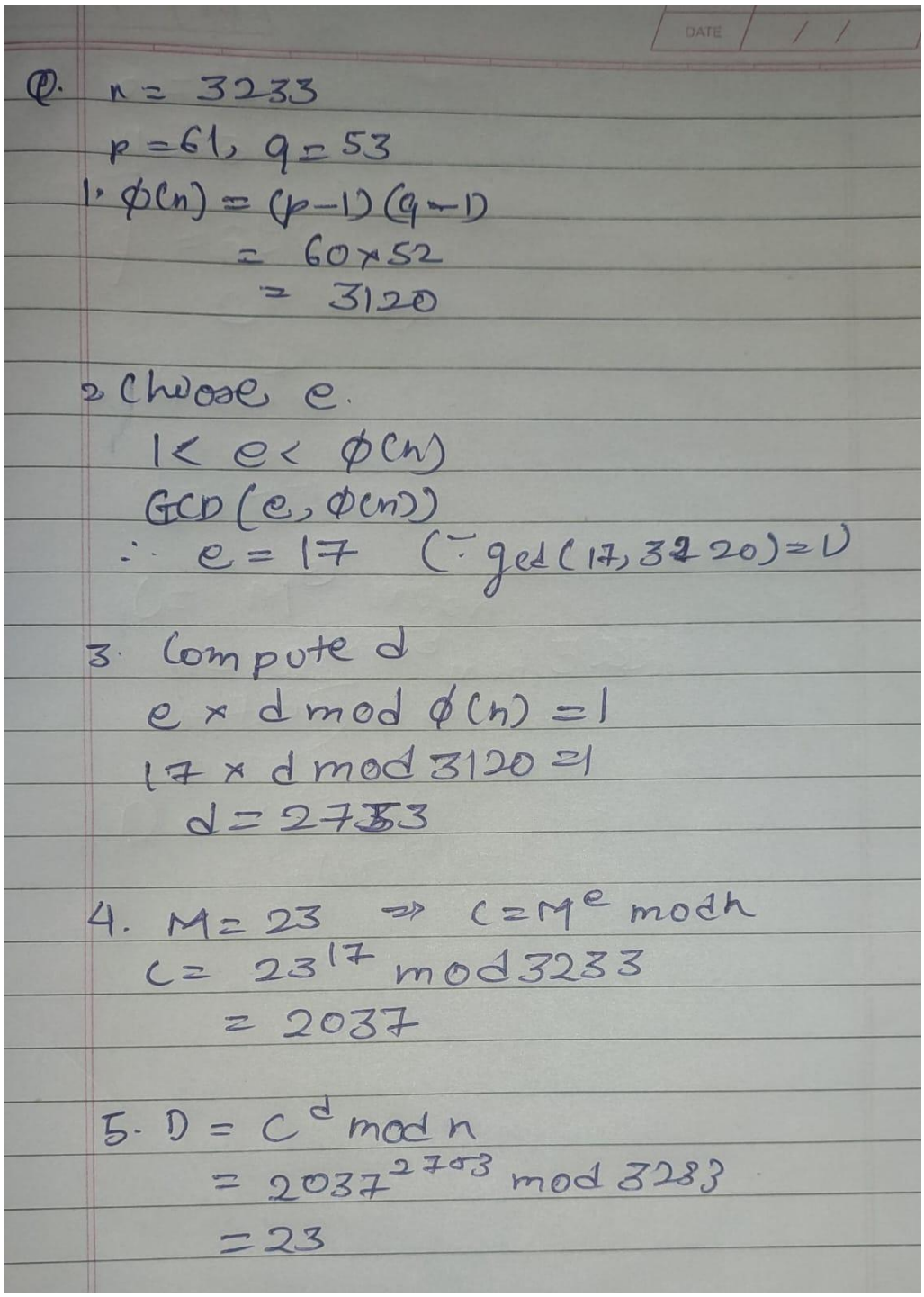
RSA Cryptanalysis Code and Output (Cyclic Attack):

```
def rsa_encrypt(m, n, e):
    return pow(m, e, n)

def cyclic_attack(n, e, start_message):
    print(f"Cyclic Attack starting from message {start_message}")
    seen = {}
    m = start_message
    step = 0

    while m not in seen:
        seen[m] = step
        c = rsa_encrypt(m, n, e)
        print(f"Step {step}: {m} -> {c}")
        m = c
        step += 1

    cycle_start = seen[m]
    cycle_length = step - cycle_start
    print(f"\nCycle detected starting at step {cycle_start} with length
{cycle_length}")
    print(f"Cycle values: {list(seen.keys())[cycle_start:]}")

p = 61
q = 53
n = p * q
e = 17
start_message = 23

cyclic_attack(n, e, start_message)
```

```
Cyclic Attack starting from message 23
Step 0: 23 -> 2037
Step 1: 2037 -> 2249
Step 2: 2249 -> 765
Step 3: 765 -> 23

Cycle detected starting at step 0 with length 4
Cycle values: [23, 2037, 2249, 765]
```

Post Lab Subjective/Objective type Questions:

1. **Solve one example using RSA algorithm.**



Q. n = 3233
p = 61, q = 53
1. $\phi(n) = (p-1)(q-1)$
$= 60 \times 52$
$= 3120$

2. Choose e.
$1 < e < \phi(n)$
GCD $(e, \phi(n))$
$\therefore e = 17$ ($\because$ gcd(17, 3120) = 1)

3. Compute d
$e \times d \bmod \phi(n) = 1$
$17 \times d \bmod 3120 = 1$
$d = 2753$

4. $M = 23 \Rightarrow C = M^e \bmod n$
$C = 23^{17} \bmod 3233$
$= 2037$

5. $D = C^d \bmod n$
$= 2037^{2753} \bmod 3233$
$= 23$

2. **Write Applications of RSA.**

   i. Used to encrypt sensitive information sent over insecure channels like the internet.

   ii. Ensures authentication, integrity, and non-repudiation by allowing senders to sign messages and receivers to verify them.

   iii. Helps securely share symmetric encryption keys over insecure networks.

   iv. Provides encryption and digital signatures for secure email communication.

   v. Used to verify user or device identity in secure systems.

   vi. Used for transaction signing and verifying authenticity in blockchain networks.

   vii. Protects software from tampering and validates the source by signing executables.

3. **Comment on the strengths and weaknesses of RSA.**

   Strengths of RSA:

1. Based on the mathematical difficulty of factoring large integers, making it secure when large key sizes (≥2048 bits) are used.
2. Can be used both for encrypting data and for digital signatures.
3. Eliminates the need to securely share keys beforehand.
4. Supported in many security protocols (SSL/TLS, PGP, SSH).

   Weaknesses of RSA:

1. Encryption and decryption are computationally intensive compared to symmetric algorithms like AES.
2. Requires very large keys for strong security, which increases computation time and memory usage.
3. Small primes or predictable random numbers make RSA easy to break.
4. Typically used to encrypt small amounts of data (like symmetric keys), not bulk data.

Conclusion:

We have successfully implemented the RSA algorithm for encryption and decryption, understanding its working principles involving key generation, modular arithmetic and prime factorization. We also performed RSA cryptanalysis using a cyclic attack.