INSTITUTE OF COMPUTER TECHNOLOGY

B-TECH COMPUTER SCIENCE ENGINEERING 2025-26

SUBJECT:-CRYPTOGRAPHY

NAME: Rahul Prajapati

ENRLL. NO: 23162171020

BRANCH: CYBER SECURITY

BATCH: 52

PRACTICAL_10

Aim: To demonstrate the Diffie–Hellman key-agreement protocol using notation from the lecture slides — generate private keys (XA, XB) for USER A and USER B, compute public values (YA, YB) using base α modulo prime q, and verify that both users derive the same shared key (KA = KB).

CODE:

practical10_1.py > ...

```python
import argparse
import sys

def modexp(base: int, exponent: int, modulus: int) -> int:
    """Efficient modular exponentiation (binary exponentiation)."""
    result = 1
    base %= modulus
    while exponent > 0:
        if exponent & 1:
            result = (result * base) % modulus
        exponent >>= 1
        base = (base * base) % modulus
    return result

def is_probable_prime(n: int) -> bool:
    """Simple deterministic checks for small n. For classroom/demo use only."""
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False
    i = 3
    while i * i <= n:
        if n % i == 0:
            return False
        i += 2
    return True

def main(argv=None):
    parser = argparse.ArgumentParser(description="Diffie–Hellman demo")
    parser.add_argument("--q", type=int, help="prime modulus q")
    parser.add_argument("--alpha", type=int, help="primitive root alpha modulo q")
    parser.add_argument("--XA", type=int, help="private key for user A")
    parser.add_argument("--XB", type=int, help="private key for user B")
    args = parser.parse_args(argv)
```

```python
    try:
        if args.q is None:
            q = int(input("Enter a prime modulus q: "))
        else:
            q = args.q
        if args.alpha is None:
            alpha = int(input("Enter a primitive root modulo q (alpha): "))
        else:
            alpha = args.alpha
        if args.XA is None:
            XA = int(input("USER A, enter your private key (XA): "))
        else:
            XA = args.XA
        if args.XB is None:
            XB = int(input("USER B, enter your private key (XB): "))
        else:
            XB = args.XB
    except ValueError:
        print("Invalid input. Please enter integer values.")
        return 2

    if not is_probable_prime(q):
        print("Warning: q does not appear to be prime. Choose a prime for real use.")

    if not (1 <= XA <= q - 2) or not (1 <= XB <= q - 2):
        print("Private keys must be in the range 1 to q-2.")
        return 3

    YA = modexp(alpha, XA, q)
    YB = modexp(alpha, XB, q)

    print(f"USER A public value (YA): {YA}")
    print(f"USER B public value (YB): {YB}")

    KA = modexp(YB, XA, q)
    KB = modexp(YA, XB, q)

    print(f"USER A computed shared key (KA): {KA}")
    print(f"USER B computed shared key (KB): {KB}")

    if KA == KB:
        print("Key agreement successful: KA == KB")
        return 0
    else:
        print("Key agreement failed: KA != KB")
        return 4

if __name__ == "__main__":
    sys.exit(main())
```

OUTPUT:

```
PS C:\Users\Hp\OneDrive\Desktop\SEM_05\Cryptography\Practicals_source_code> C:/Users/Hp/OneDrive/Desktop/SEM_05/Cryptography/Practicals_source_code/.venv/Scripts/python.exe .\practical10_1.py
Enter a prime modulus q: 97
Enter a primitive root modulo q (alpha): 131
Enter a primitive root modulo q (alpha): 131
USER A, enter your private key (XA): 6
USER A, enter your private key (XA): 6
USER B, enter your private key (XB): 54
USER A public value (YA): 70
USER B public value (YB): 27
USER A computed shared key (KA): 64
USER B computed shared key (KB): 64
Key agreement successful: KA == KB
USER A public value (YA): 70
USER B public value (YB): 27
USER A computed shared key (KA): 64
USER B computed shared key (KB): 64
Key agreement successful: KA == KB
USER B public value (YB): 27
USER A computed shared key (KA): 64
USER B computed shared key (KB): 64
Key agreement successful: KA == KB
USER B computed shared key (KB): 64
Key agreement successful: KA == KB
Key agreement successful: KA == KB
PS C:\Users\Hp\OneDrive\Desktop\SEM_05\Cryptography\Practicals_source_code>
```