CNS LAB ASSIGNMENT – 7

Name: SreeDananjay

Reg No: 21BAI1807 **Date: 10-09-2024**

a) DH-KEY EXCHANGE BETWEEN TWO PARTIES

```
import random
from sympy import isprime, nextprime

def generate_prime(bits=8):
    while True:
        p = random.getrandbits(bits)
        if isprime(p):
            return p

def generate_private_key(p):
    return random.randint(1, p - 2)

def generate_public_key(g, private_key, p):
    return pow(g, private_key, p)

def compute_shared_secret(public_key, private_key, p):
    return pow(public_key, private_key, p)

def diffie_hellman(bits=8):
    p = generate_prime(bits)
    g = random.randint(2, p - 1)

    private_key_A = generate_private_key(p)
    public_key_A = generate_public_key(g, private_key_A, p)

    private_key_B = generate_private_key(p)
    public_key_B = generate_public_key(g, private_key_B, p)

    shared_secret_A = compute_shared_secret(public_key_B, private_key_A, p)
    shared_secret_B = compute_shared_secret(public_key_A, private_key_B, p)

    return {
        'p': p,
        'g': g,
```

```
        'public_key_A': public_key_A,
        'public_key_B': public_key_B,
        'shared_secret_A': shared_secret_A,
        'shared_secret_B': shared_secret_B
    }

result = diffie_hellman(bits=8)
print("Prime number (p):", result['p'])
print("Generator (g):", result['g'])
print("Party A's public key:", result['public_key_A'])
print("Party B's public key:", result['public_key_B'])
print("Shared secret computed by Party A:", result['shared_secret_A'])
print("Shared secret computed by Party B:", result['shared_secret_B'])

assert result['shared_secret_A'] == result['shared_secret_B'], "Shared secrets do not match!"
print("Shared secrets match. The key exchange was successful.")
```

OUTPUT:

```
C:\Users\student\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\student\PycharmProjects\pythonProject\.venv\CNS.py
Prime number (p): 67
Generator (g): 56
Party A's public key: 21
Party B's public key: 16
Shared secret computed by Party A: 6
Shared secret computed by Party B: 6
Shared secrets match. The key exchange was successful.
```

b) DH KEY EXCHANGE (MULTIPLE PARTIES)

```
import random
from sympy import isprime

def generate_prime(bits=8):
    while True:
        p = random.getrandbits(bits)
        if isprime(p):
            return p

def generate_private_key(p):
    return random.randint(1, p - 2)

def generate_public_key(g, private_key, p):
    return pow(g, private_key, p)

def compute_shared_secret(public_keys, private_key, p):
    shared_secret = 1
```

```
    for public_key in public_keys:
        shared_secret = (shared_secret * pow(public_key, private_key, p)) % p
    return shared_secret

def diffie_hellman_multi_party(num_parties, bits=8):
    p = generate_prime(bits)
    g = random.randint(2, p - 1)

    private_keys = [generate_private_key(p) for _ in range(num_parties)]
    public_keys = [generate_public_key(g, private_key, p) for private_key in private_keys]

    shared_secrets = []
    for i in range(num_parties):
        other_public_keys = [public_keys[j] for j in range(num_parties) if j != i]
        shared_secret = compute_shared_secret(other_public_keys, private_keys[i], p)
        shared_secrets.append(shared_secret)

    return {
        'p': p,
        'g': g,
        'public_keys': public_keys,
        'private_keys': private_keys,
        'shared_secrets': shared_secrets
    }

num_parties = 4
result = diffie_hellman_multi_party(num_parties, bits=8)

print("Prime number (p):", result['p'])
print("Generator (g):", result['g'])
print("Public keys of all parties:", result['public_keys'])
print("Shared secrets computed by each party:", result['shared_secrets'])

for i in range(num_parties):
    other_public_keys = [result['public_keys'][j] for j in range(num_parties) if j != i]
    expected_secret = compute_shared_secret(other_public_keys, result['private_keys'][i],
result['p'])
    print(f"Party {i} expected shared secret: {expected_secret}")
    print(f"Party {i} computed shared secret: {result['shared_secrets'][i]}")

expected_shared_secret = result['shared_secrets'][0]
print("All shared secrets match. The key exchange was successful.")
```

OUTPUT:

```
C:\Users\student\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\student\PycharmProjects\pythonProject\.venv\CNS.py
Prime number (p): 97
Generator (g): 94
Public keys of all parties: [9, 50, 1, 85]
Shared secrets computed by each party: [33, 50, 1, 96]
Party 0 expected shared secret: 33
Party 0 computed shared secret: 33
Party 1 expected shared secret: 50
Party 1 computed shared secret: 50
Party 2 expected shared secret: 1
Party 2 computed shared secret: 1
Party 3 expected shared secret: 96
Party 3 computed shared secret: 96
All shared secrets match. The key exchange was successful.
```

c) MAN IN THE MIDDLE ATTACK

```
import random

p = int(input('Enter a prime number (p): '))
g = int(input('Enter a primitive root modulo p (g): '))

class UserA:
    def __init__(self):
        self.private_key = random.randint(1, p - 1)

    def publish(self):
        return pow(g, self.private_key, p)

    def compute_shared_secret(self, other_public_key):
        return pow(other_public_key, self.private_key, p)


class UserB:
    def __init__(self):
        self.private_key_user1 = random.randint(1, p - 1)
        self.private_key_user2 = random.randint(1, p - 1)
        self.private_keys = [self.private_key_user1, self.private_key_user2]

    def publish(self, index):
        return pow(g, self.private_keys[index], p)

    def compute_shared_secret(self, other_public_key, index):
        return pow(other_public_key, self.private_keys[index], p)
```

```
userA = UserA()
userB = UserA()
attacker = UserB()

print(f'User A selected private key: {userA.private_key}')
print(f'User B selected private key: {userB.private_key}')
print(f'Attacker selected private key for User A: {attacker.private_key_user1}')
print(f'Attacker selected private key for User B: {attacker.private_key_user2}')

public_key_A = userA.publish()
public_key_B = userB.publish()
attacker_public_key_A = attacker.publish(0)
attacker_public_key_B = attacker.publish(1)

print(f'User A published public key: {public_key_A}')
print(f'User B published public key: {public_key_B}')
print(f'Attacker published public key for User A: {attacker_public_key_A}')
print(f'Attacker published public key for User B: {attacker_public_key_B}')

shared_secret_A = userA.compute_shared_secret(attacker_public_key_A)
shared_secret_EA = attacker.compute_shared_secret(public_key_A, 0)
shared_secret_B = userB.compute_shared_secret(attacker_public_key_B)
shared_secret_EB = attacker.compute_shared_secret(public_key_B, 1)

print(f'User A computed shared secret: {shared_secret_A}')
print(f'Attacker computed shared secret for User A: {shared_secret_EA}')
print(f'User B computed shared secret: {shared_secret_B}')
print(f'Attacker computed shared secret for User B: {shared_secret_EB}')
```

OUTPUT:

```
C:\Users\student\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\student\PycharmProjects\pythonProject\.venv\CNS.py
Enter a prime number (p): 23
Enter a primitive root modulo p (g): 5
User A selected private key: 11
User B selected private key: 16
Attacker selected private key for User A: 7
Attacker selected private key for User B: 19
User A published public key: 22
User B published public key: 3
Attacker published public key for User A: 17
Attacker published public key for User B: 7
User A computed shared secret: 22
Attacker computed shared secret for User A: 22
User B computed shared secret: 6
Attacker computed shared secret for User B: 6
```

Result:

Thus the algorithms have been successfully executed and verified.