```python
import socket
import random
from Crypto.Cipher import DES
from Crypto.Util.Padding import unpad
from Crypto.Random import get_random_bytes  # To generate a secure random IV

# Diffie-Hellman Parameters
p = 23  # A small prime number
g = 5   # A primitive root modulo p

# Step 1: Server generates private key and public key
private_key = random.randint(1, p-1)
public_key = pow(g, private_key, p)

# Create server socket
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('0.0.0.0', 12345))  # Use '0.0.0.0' to accept connections from
any IP
server.listen(1)

print("Waiting for connection...")
client_socket, client_address = server.accept()
print(f"Connection established with {client_address}")

# Step 2: Send server's public key to client
client_socket.send(str(public_key).encode())

# Step 3: Receive client's public key
client_public_key = int(client_socket.recv(1024).decode())

# Step 4: Calculate the shared secret
shared_secret = pow(client_public_key, private_key, p)

# Step 5: Generate encryption key (hash the shared secret)
encryption_key = str(shared_secret).zfill(8).encode()[:8]

# DES decryption setup
# CBC mode requires an IV
iv = get_random_bytes(DES.block_size)  # Generate a random 8-byte IV

cipher = DES.new(encryption_key, DES.MODE_CBC, iv)

# Step 6: Receive and decrypt the message
# Receive the IV from the client first (as it needs to be sent with the
message)
received_iv = client_socket.recv(DES.block_size)

# Now, set the cipher to the received IV
```

```python
cipher = DES.new(encryption_key, DES.MODE_CBC, received_iv)

# Receive the encrypted message
encrypted_message = client_socket.recv(1024)

# Ensure unpadding after decryption
try:
    decrypted_message = unpad(cipher.decrypt(encrypted_message),
DES.block_size).decode('utf-8')
    print(f"Decrypted message from client: {decrypted_message}")
except ValueError as e:
    print(f"Error during unpadding or decryption: {e}")
except UnicodeDecodeError as e:
    print(f"Error during decoding: {e}")

# Close connection
client_socket.close()
server.close()
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

(venv) C:\Users\Student\Documents\cns lab>python server.py
Waiting for connection...
Connection established with ('172.16.77.53', 50463)
Decrypted message from client: Hello, Server! This is a secret message.

(venv) C:\Users\Student\Documents\cns lab>
```

```python
import socket
import random
from Crypto.Cipher import DES
from Crypto.Util.Padding import pad
from Crypto.Random import get_random_bytes  # To generate a secure random IV

# Diffie-Hellman Parameters
p = 23  # A small prime number
g = 5   # A primitive root modulo p

# Step 1: Client generates private key and public key
private_key = random.randint(1, p-1)
public_key = pow(g, private_key, p)

# Create client socket
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('172.16.77.52', 12345))
```

```python
# Step 2: Receive server's public key
server_public_key = int(client.recv(1024).decode())

# Step 3: Send client's public key to server
client.send(str(public_key).encode())

# Step 4: Calculate the shared secret
shared_secret = pow(server_public_key, private_key, p)

# Step 5: Generate encryption key (hash the shared secret)
encryption_key = str(shared_secret).zfill(8).encode()[:8]

# DES encryption setup
# CBC mode requires an IV
iv = get_random_bytes(DES.block_size)  # Generate a random 8-byte IV
cipher = DES.new(encryption_key, DES.MODE_CBC, iv)

# Encrypt message
message = "Hello, Server! This is a secret message."
padded_message = pad(message.encode(), DES.block_size)
encrypted_message = cipher.encrypt(padded_message)

# Step 6: Send IV and encrypted message to server
client.send(iv)  # Send the IV first
client.send(encrypted_message)  # Send the encrypted message

client.close()
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

(venv) C:\Users\Student\Desktop\assignmnet 3 b>python client.py

(venv) C:\Users\Student\Desktop\assignmnet 3 b>
```