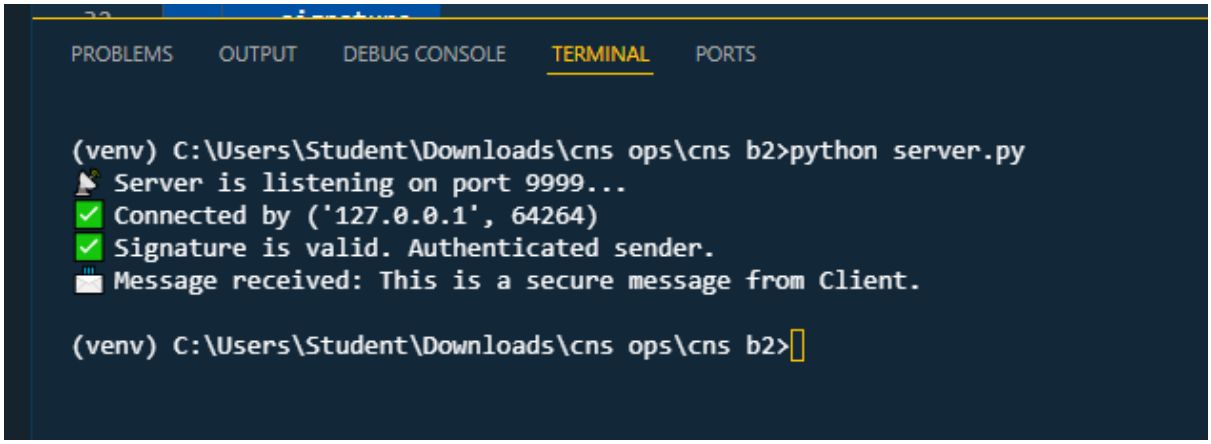
Server.py

```python
import socket
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding

# Create the server socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 9999))
server_socket.listen(1)
print("📡 Server is listening on port 9999...")

conn, addr = server_socket.accept()
print(f"✅ Connected by {addr}")

# Receive data
data = b""
while True:
    chunk = conn.recv(4096)
    if not chunk:
        break
    data += chunk

try:
    # Split the received data
    public_pem, rest = data.split(b"<ENDKEY>")
    message, signature = rest.split(b"<ENDMSG>")

    # Load the public key
    public_key = serialization.load_pem_public_key(public_pem)

    # Verify the signature
    public_key.verify(
        signature,
        message,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )

    print("✅ Signature is valid. Authenticated sender.")
    print(f"📩 Message received: {message.decode()}")

except Exception as e:
    print("❌ Signature verification failed!")
    print("Error:", str(e))
```

```
# Clean up
conn.close()
server_socket.close()
```



Client.py

```
import socket
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.hazmat.primitives.asymmetric import rsa, padding

# Generate RSA Key Pair
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
public_key = private_key.public_key()

# Serialize the public key to send to the server
public_pem = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)

# Message to be signed and sent
message = b"This is a secure message from Client."

# Create the signature using the private key
signature = private_key.sign(
    message,
    padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),
        salt_length=padding.PSS.MAX_LENGTH
    ),
    hashes.SHA256()
)

# Connect to the server
```

```python
server_ip = '127.0.0.1'  # Replace with actual server IP
server_port = 9999       # Port must match the server's listening port

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((server_ip, server_port))

# Prepare data: public key, message, signature separated by special markers
data = public_pem + b"<ENDKEY>" + message + b"<ENDMSG>" + signature

# Send the combined data
client_socket.sendall(data)
print("📤 Sent public key, message, and signature to server.")

# Close the socket
client_socket.close()
```