Information Security

First, we create a client side user interface. The client enters their message. The client enters their key and their message.

Here we give the client the option to choose on how they want to encrypt their message. If the client chooses symmetric, they get the option of AES and SHA256. If the client chooses asymmetric, they get the option of RSA.

Secure Message Encryption

Enter your message:

Enter a key:

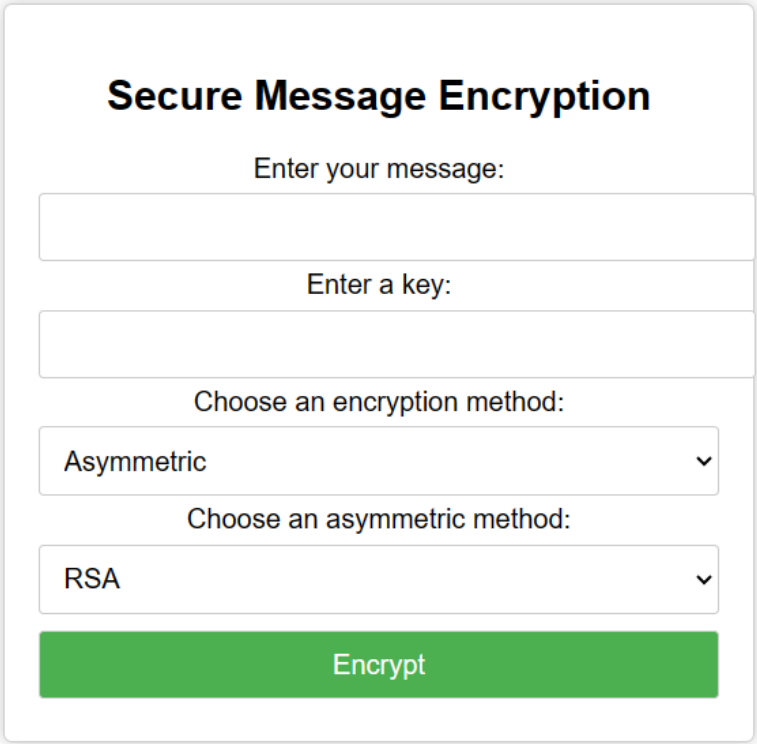
Choose an encryption method:

Symmetric

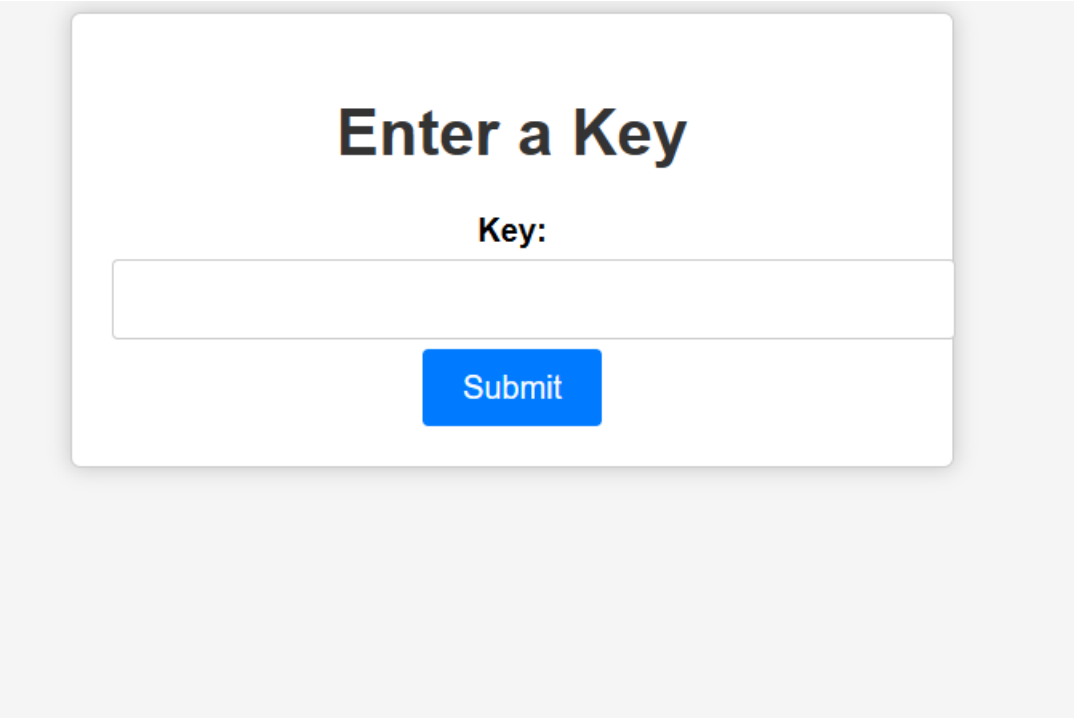
Choose a symmetric method:

AES

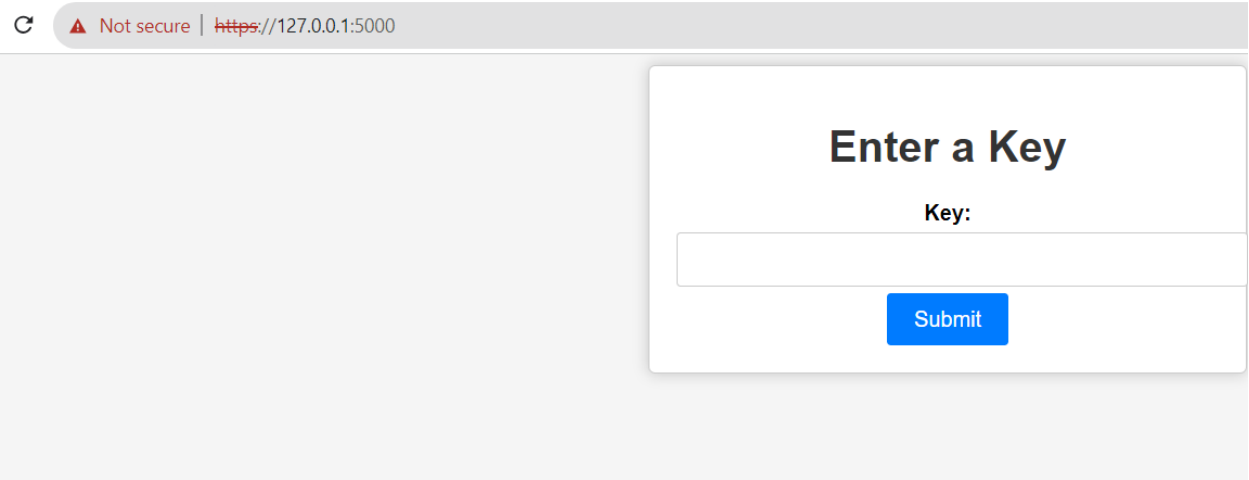
Encrypt



The server sees one interface that asks them to enter their key.



Here we have used SSL on the server side. We have used cert.pem and key.kem. Now when we access the page it calls our page not secure since these keys cannot be authenticated by the server.



Now the server and the client say hello to each other.

client.py 2

C: > Users > mayha > OneDrive > Desktop > A2-info > client.py > say_hello_to_server

```python
asymmetricMethod = ""  # Define asymmetricMethod at the global level
Chosenmethod = ""  # Define Chosenmethod at the global level
key = 10

@app.route('/')
def home():
    return render_template('user.html')

@app.route('/say_hello_to_server')
def say_hello_to_server():
    # Send a "hello" message to the server
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 * Debugger PIN: 354-254-525
hi mayhan
Received Message: hi
the key you chose is  10
hi mayhan
Received Message: hi
the key you chose is  10
PS C:\Users\mayha> & C:/Users/mayha/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/mayha/
rive/Desktop/A2-info/client.py
 * Serving Flask app 'client'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
ver instead.
 * Running on https://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 354-254-525
Server said Hello
127.0.0.1 - - [27/Oct/2023 09:44:11] "GET /say_hello_to_server HTTP/1.1" 200 -
127.0.0.1 - - [27/Oct/2023 09:44:11] "GET /favicon.ico HTTP/1.1" 404 -
```

Ln 33, Col 82 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (Microsoft Store) Go Live

server.py

C: > Users > mayha > OneDrive > Desktop > A2-info > server.py > ...

```python
app = Flask(__name__)
global key
key = 10
@app.route('/')
def home():
    return render_template('server.html')

@app.route('/receive_hello', methods=['GET'])
def receive_hello():
    # The server receives the "hello" message from the client
    print("Client said hello")
    return "Server received message: 'hello'"

@app.route('/process', methods=['POST'])
def process():
    global key
    key = request.form['key']
    return f'You entered the key: {key}'
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 * Serving Flask app 'server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a p
roduction WSGI server instead.
 * Running on https://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 354-254-525
 * Detected change in 'c:\\Users\\mayha\\OneDrive\\Desktop\\A2-info\\client.py', reloadi
ng
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 354-254-525
Client said hello
127.0.0.1 - - [27/Oct/2023 09:44:11] "GET /receive_hello HTTP/1.1" 200 -
```

Ln 15, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (Microsoft Store) Go Live

Now the client will tell the server what kind of encryption method it will be using.

Based on the send chosen method the receive chosen method is called on the server side. The server will receive the method that was chosen.

```python
    #response = requests.post('https://localhost:5001/receive_chosen_method', json=data, verify=False)
    data = {'method': Chosenmethod, 'key': serialized_client_public_key.decode('utf-8')}
    response = requests.post('https://localhost:5001/receive_chosen_method', json=data, verify=False)

    if response.status_code == 200:
        server_response = response.json()
        server_key_path = os.path.join(keys_directory, 'server_public_key_received.pem')

        # Write the received key to a file
        #with open(server_key_path, 'wb') as key_file:
        with open(server_key_path, 'wb') as key_file:
            key_file.write(server_response['key'].encode('utf-8'))

        # Read the key from the file
        server_key_path=r"C:\Users\mayha\OneDrive\Desktop\7thSemester\IS\A2-info\Serverkeys\server_public_key.pem"
        server_public_key = deserialize_public_key(server_key_path)

        # Perform the key exchange
        shared_key = client_private_key.exchange(server_public_key)
        shared_key_bytes = shared_key.to_bytes((shared_key.bit_length() + 7) // 8, byteorder='big')

        # Write the shared key to a file
        shared_key_file_path = r"C:\Users\mayha\OneDrive\Desktop\7thSemester\IS\A2-info\Serverkeys\shared_key.pem"
        with open(shared_key_file_path, 'wb') as key_file:
            key_file.write(shared_key_bytes)
```

This is also where we are sending and receiving the keys. Based the method that was chosen, the keys are generated and written in the file. The server side will then read that key.

```
PS C:\Users\mayha\OneDrive\Desktop\7thSemester\IS\A2-info> python server.py
before
i got here
 * Serving Flask app 'server'
 * Debug mode: on
 * Debugger is active!
 * Debugger PIN: 354-254-525
the chosen method was  AES
the key you entered was -----BEGIN PUBLIC KEY-----
MIICJTCCARcGCSqGSIb3DQEDATCCAQgCggEBAI1omhI7hNIJhm6FXjyZxQ6H4pDL
ApQDVwcUw48UWoQxxdpH1262UtYy3E6zgua1XoOXwGUQbN4/0JIJITpzlaUDal7v
J8wZcu4ohCeUzx/zNc+CQAj18mlOyl9+8oj+oor8nsXd3Me9D4qkzU+ojBc89+9F
pMH0WO0QGtfhyBY7JtoR5Lcphw6nn/VDtzasBnEHVFBeBx7LS18y0kvm8/31ms5O
mXTtxfk82UY8JkKvesh2wYUtY2BZFfI4c3HshqKqJbBg8iKkki4+M9KMhMIHLOAl
z3u0wYTWFW+q0CZHyUWlqjjWMhqDLlku4cUmxlFgY08Wz/9hUKZaGaKhaS8CAQID
ggEGAAKCAQEAjBemX3v/OgiCvTcIv8kqm4O+/D2EwUfyO/qjS/g4fi8HFb7opsfC
k0IH9kUeq1lrnZRN8pJzRSpTRS6tlGV2Gh3Cx5GA0Khdo0hk/ubKm6cuHYCebaj6
khqI48juiA9go7BLcwNx57IvYtrPrPMuFAM/6Fw0jtpCZUtvOcSt/qymO5hCRfZl
MtPqxF5B+3nREaHGeKwhE9c+0NdBQc5c+fo78g426e/xZgHZv5wjgCOzAnX2aKd6
c1Z8MG3g29yU80XMmVQGd7YJK12PYwPcHNrzasrZ6HxUB0ncm3nfmNzGYNTUp9gA
DavX697sGAOBmdFbeOYHQn2EqzpgXizqrg==
-----END PUBLIC KEY-----
```