

# CN Project Documentation

## Secure Cloud-Based File Storage System

### Computer Networks Project

#### Project Details

- **Title:** Secure Cloud-Based File Storage System (Client-Server Network)
- **Project Members:**
  - Adyansh Aggarwal (PES1UG23AM028)
  - Akash Madisetty (PES1UG23AM035)

#### Problem Statement

Development of an encrypted file transfer and storage system that ensures secure communication between clients and servers while maintaining data confidentiality.

#### Project Aim

To build a client-server system where multiple clients can **store encrypted files on Google Drive** using **AES-level encryption**, ensuring secure transmission and storage of sensitive data. The system implements **TCP checksum** for data integrity verification and **SSL/TLS encryption** for secure communication channels, providing multiple layers of security for file transfers.

#### System Architecture

##### 1. Server Components

- **server.py:** Main server implementation
- **run\_server.py:** Server execution script
- **Server.crt:** SSL certificate
- **Server.key:** SSL private key
- **generate\_ssl.py :** Generates the SSL key and certificate for the server
- **encryption.py:** AES encryption/decryption implementation
- **gdrive.py:** Google Drive API integration
- **test\_encryption.py:** Encryption testing module
- **Downloads directory:** Temporary storage for incoming files

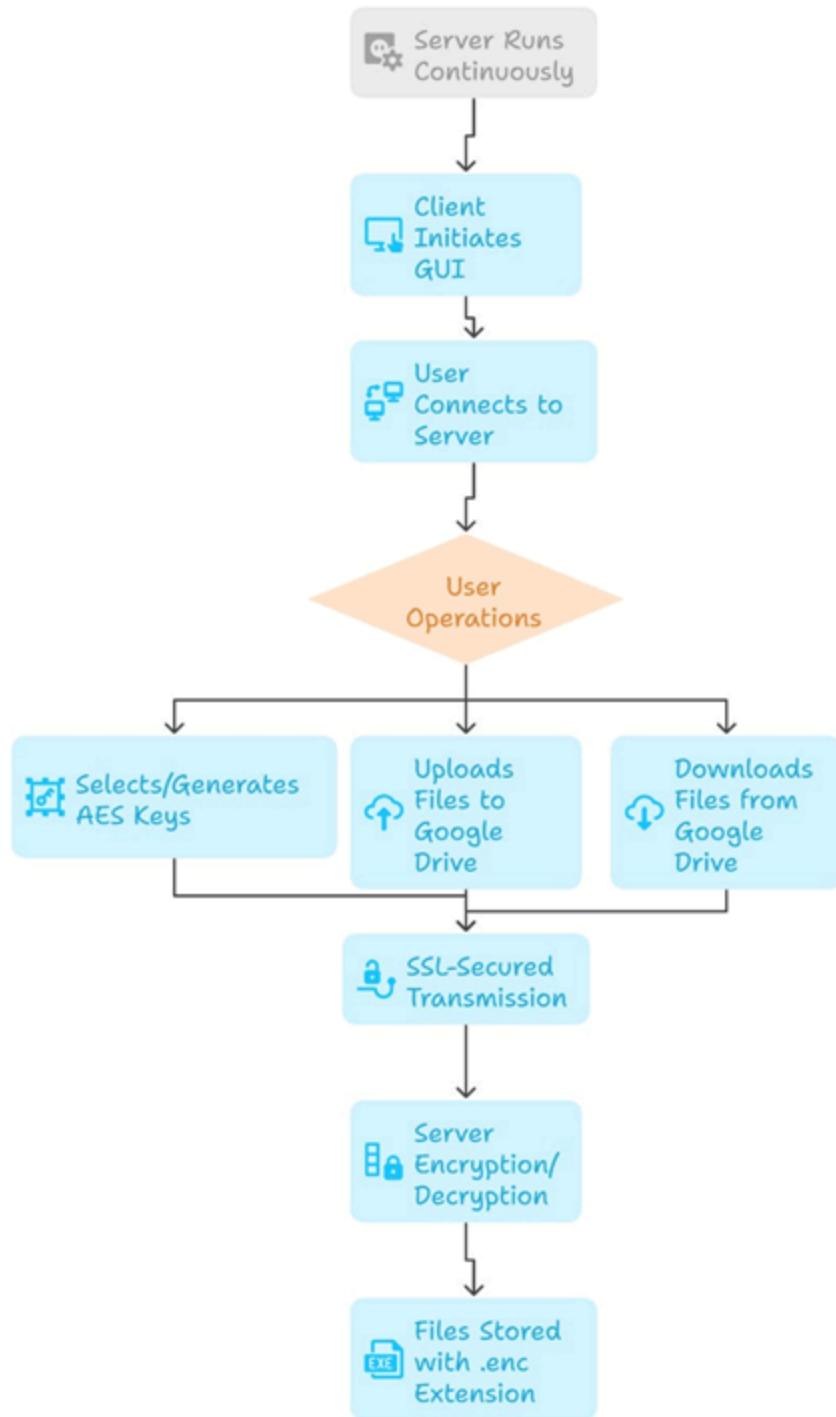
- **Uploads directory:** Temporary storage for outgoing files

## 2. Client Components

- **gui.py:** User interface implementation
- **client.py:** Client-side logic
- **Server.crt:** Server's SSL certificate
- **run\_client.py:** Client execution script

## Process Flow

## Secure File Management Process



## Security Features

### 1. SSL/TLS Encryption:

- Secure Socket Layer implementation
- Server authentication through certificates

- Encrypted communication channel

## 2. AES Encryption:

- File-level encryption using AES
- Local key storage on client side
- Protection against unauthorized access

## 3. Data Integrity:

- TCP checksum verification
- Protection against data corruption
- Secure file transfer protocols

# Key Security Considerations

## 1. Client-Side Key Management:

- AES keys stored locally on client machines
- **Loss of keys results in permanent data inaccessibility**
- No server-side key storage

## 2. Server Security:

- No file storage on server
- Dedicated encryption/decryption operations
- SSL certificate validation

## 3. Data Protection:

- Single encrypted copy in Google Drive
- Corrupted data handling
- Secure transmission protocols

# Technical Stack

## 1. Core Libraries:

- Socket library for network communication
- Crypto library for AES implementation
- Google Drive API and Google Auth libraries
- Cryptography and OpenSSL libraries

## 2. User Interface:

- PyQt5 for GUI implementation
- Intuitive user experience
- Secure credential management

# File Structure

```
Project Root/
├── Server/
│   ├── server.py
│   ├── run_server.py
│   ├── Server.crt
│   ├── Server.key
│   ├── encryption.py
│   ├── gdrive.py
│   ├── generate_ssl.py
│   ├── test_encryption.py
│   ├── Downloads/
│   └── Uploads/
└── Client/
    ├── gui.py
    ├── client.py
    ├── Server.crt
    └── run_client.py
```

## Security Protocols

### 1. Authentication:

- SSL certificate validation
- Client-server handshake
- Secure key exchange

### 2. Encryption:

- AES-256 encryption standard
- SSL/TLS for transmission
- Secure key storage

### 3. Data Integrity:

- Checksum verification
- Error detection

## Error Handling

### 1. Connection Issues:

- Graceful disconnection
- Reconnection protocols - server has appropriate time out for added safety.
- Error logging

### 2. Data Corruption:

- Checksum validation
- Secure retransmission
- Data integrity verification

### 3. Key Management:

- Key validation
- Secure storage
- Access control

## Future Enhancements

### 1. Additional Features:

- Multi-factor authentication
- Key recovery mechanisms
- Enhanced file sharing capabilities

### 2. Performance Improvements:

- Parallel file processing
- Optimized encryption algorithms

### 3. Security Upgrades:

- Advanced encryption standards
- Improved key management
- Enhanced access control

## Running the Project

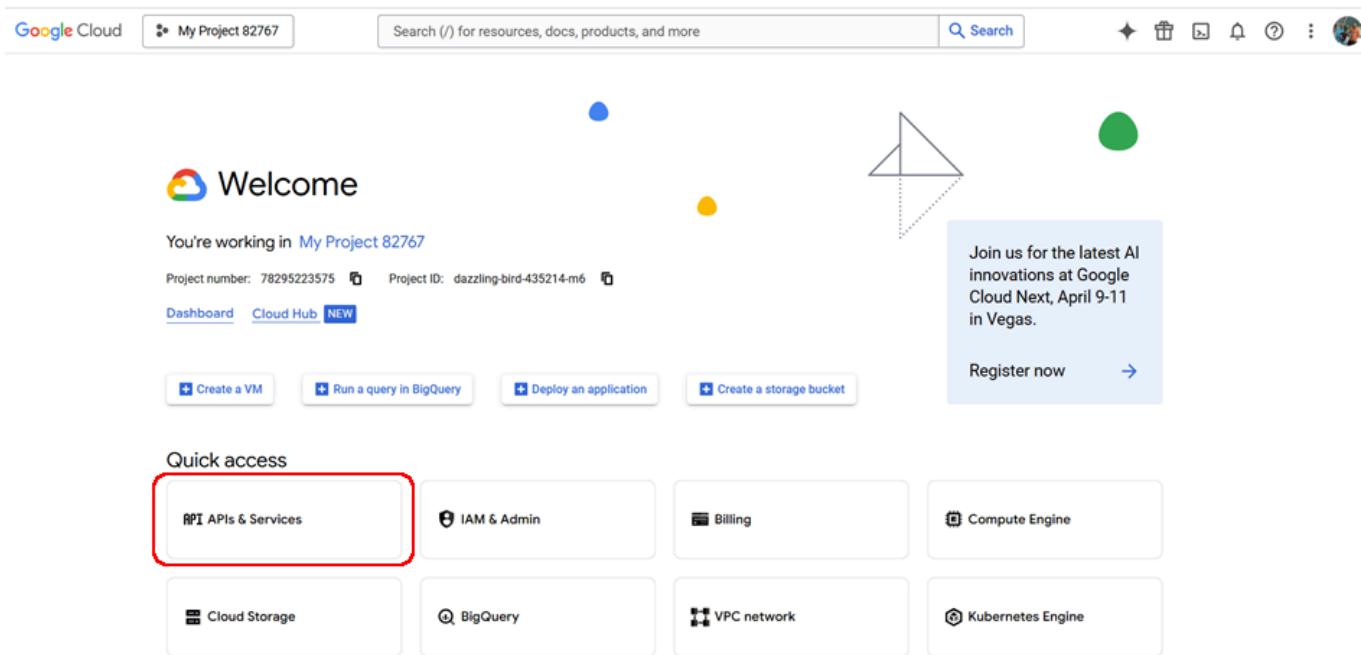
### Prerequisites

1. Python 3.x installed
2. Required Python packages:

```
pip install pycryptodome
pip install google-auth google-auth-oauthlib google-auth-httplib2 google-
api-python-client
pip install PyQt5
pip install cryptography
```

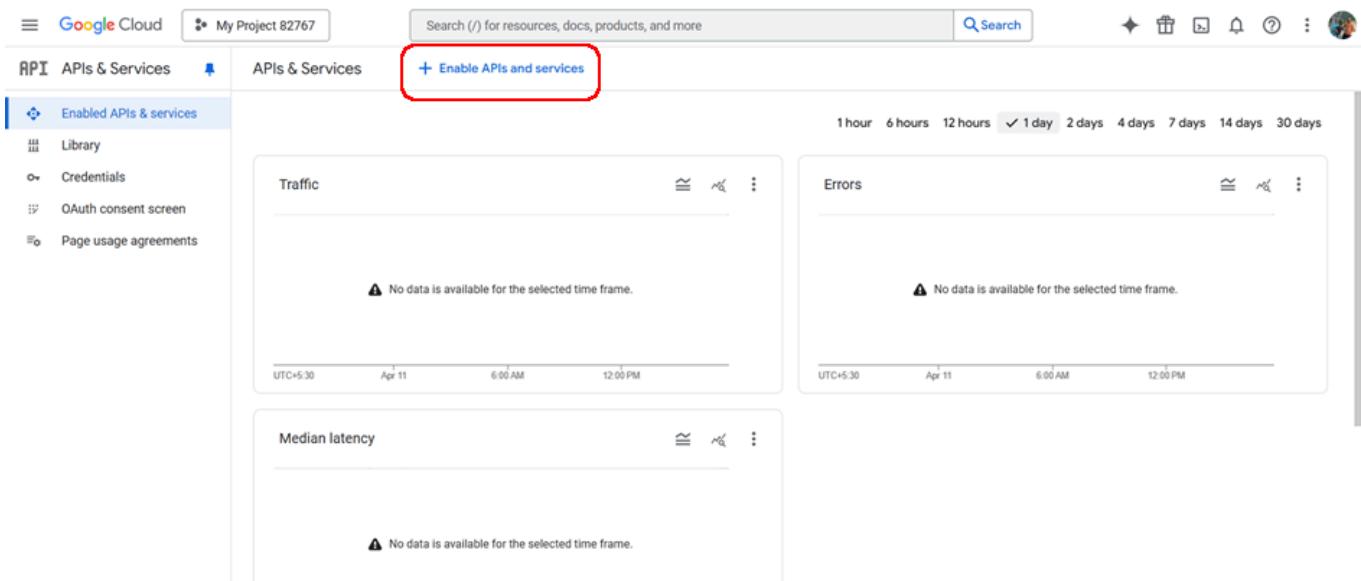
## Google Cloud Credentials

1. Create Google Cloud Console account and create a project
2. Click on APIs & Services



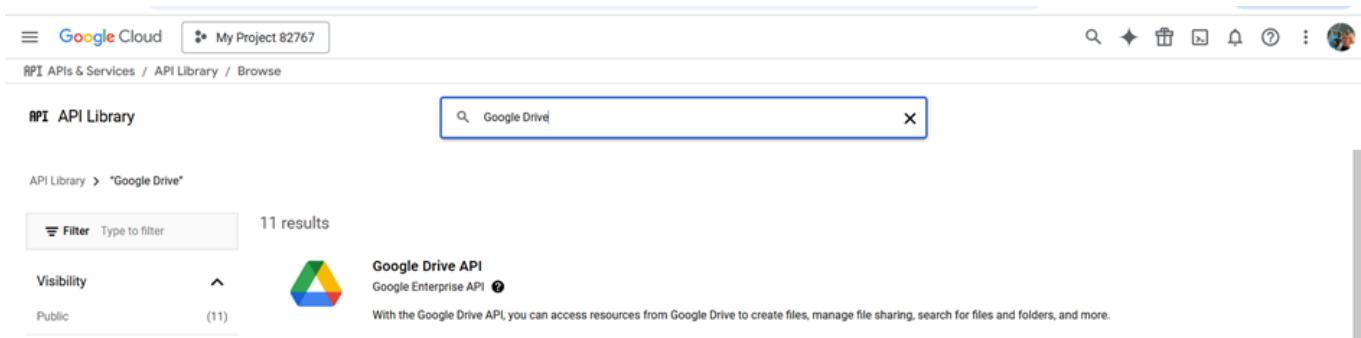
The screenshot shows the Google Cloud Welcome screen for 'My Project 82767'. At the top, there's a search bar and a navigation bar with icons for Home, My Project, Help, and Profile. Below the header, a 'Welcome' section says 'You're working in My Project 82767' with project number 78295223575 and ID dazzling-bird-435214-m6. It features four buttons: 'Create a VM', 'Run a query in BigQuery', 'Deploy an application', and 'Create a storage bucket'. To the right, there's an advertisement for Google Cloud Next and a 'Register now' button. Below these are sections for 'Quick access' and 'Recent activity'.

### 3. Click on Enable APIs and Services



The screenshot shows the 'APIs & Services' page under the 'API' tab. The left sidebar has sections for 'Enabled APIs & services' (Library, Credentials, OAuth consent screen, Page usage agreements), 'Traffic' (No data available), 'Errors' (No data available), and 'Median latency' (No data available). The main navigation bar has tabs for 'APIs & Services' and '+ Enable APIs and services', with the latter being highlighted by a red box.

### 4. Browse and select Google Drive API



The screenshot shows the 'API Library' page under the 'API' tab. The search bar contains 'Google Drive'. The results section shows 11 results for 'Google Drive API' (Google Enterprise API). A description states: 'With the Google Drive API, you can access resources from Google Drive to create files, manage file sharing, search for files and folders, and more.' There are filters for 'Visibility' (Public) and a 'Type to filter' input field.

5. Click on Enable

[Product details](#)



## Google Drive API

[Google Enterprise API](#)

Create and manage resources in Google Drive.

[ENABLE](#)

[TRY THIS API](#)

[OVERVIEW](#)

[DOCUMENTATION](#)

[SUPPORT](#)

[RELATED PRODUCTS](#)

6. This Page will appear and select create credentials

The screenshot shows the Google Cloud Platform API Services interface. On the left, a sidebar menu is open under 'Enabled APIs & services' with options like Library, Credentials, OAuth consent screen, and Page usage agreements. The main content area displays the 'API/Service Details' for the Google Drive API. At the top of this section, there is a note: 'To use this API, you may need credentials.' To the right of this note is a red rectangular box highlighting a 'Create credentials' button. Below this, the API details are shown: Service name is 'drive.googleapis.com', Type is 'Public API', and Status is 'Enabled'. To the right, there are links for Documentation (Overview, Quickstarts, API reference) and Explore (Try in API Explorer). At the bottom of the main content area, there are filters for Metrics, Quotas & System Limits, Credentials, and Drive UI Integration, along with time range dropdowns for 1 hour, 6 hours, 12 hours, 1 day, 2 days, 4 days, 7 days, 14 days, and 30 days.

7. Click on User Data and click next

## 1 Credential Type

### Which API are you using?

Different APIs use different auth platforms and some credentials can be restricted to only call certain APIs.

Select an API \*

Google Drive API

### What data will you be accessing? \*

Different credentials are required to authorize access depending on the type of data that you request. [Learn more](#)



#### User data

Data belonging to a Google user, like their email address or age. User consent required. This will create an OAuth client.



#### Application data

Data belonging to your own application, such as your app's Cloud Firestore backend. This will create a service account.

[Next](#)

8. Fill App name and e-mail (if prompted for a scope - not required to add)

## 2 OAuth Consent Screen

### App information

This shows in the consent screen, and helps end users know who you are and contact you

App name \*

! Application name must not be empty

User support email \*

! An email address must be selected

10. OAuth Client ID – Select Application type and Name

## 4 OAuth Client ID

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information. [Learn more](#) about OAuth client types.

Application type \*

Desktop app

Name \*

Desktop client 1

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

Note: It may take 5 minutes to a few hours for settings to take effect

**Create**

Cancel

11. Download the credentials and rename it as `credentials.json` and click done
12. Save this `credentials.json` to the Server

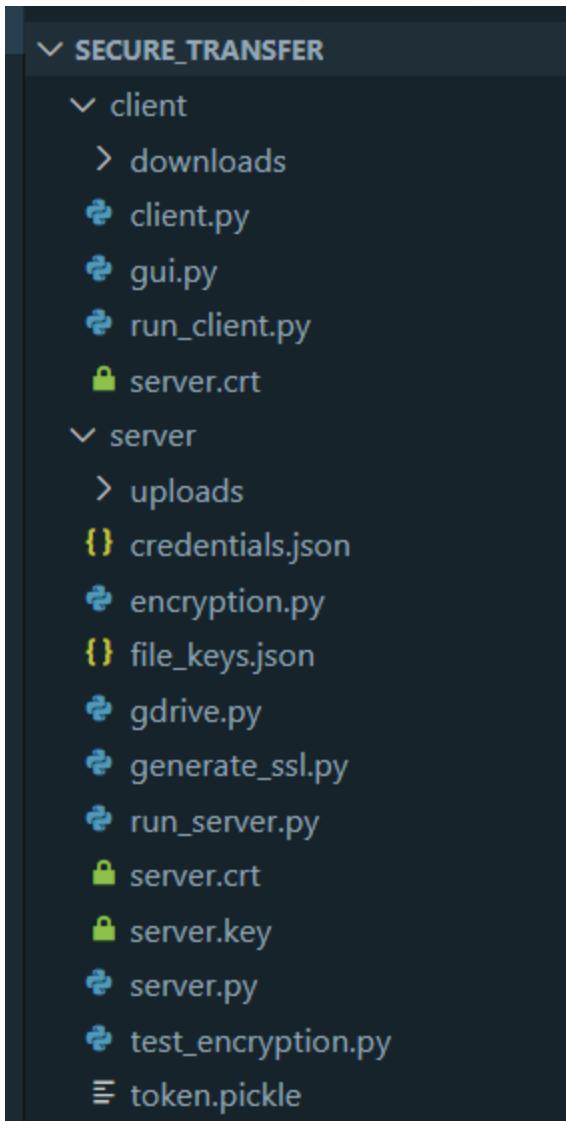
## Server Setup

1. Generate SSL certificates:

Use the below command or run the `generate_ssl.py` file in the server

```
openssl req -x509 -newkey rsa:4096 -keyout Server.key -out Server.crt -days 365 -nodes
```

2. Copy the generated `Server.crt` to the client directory



3. Start the server:

```
python run_server.py
```

- The server will start listening on the specified port
- Note down the server's IP address and port number

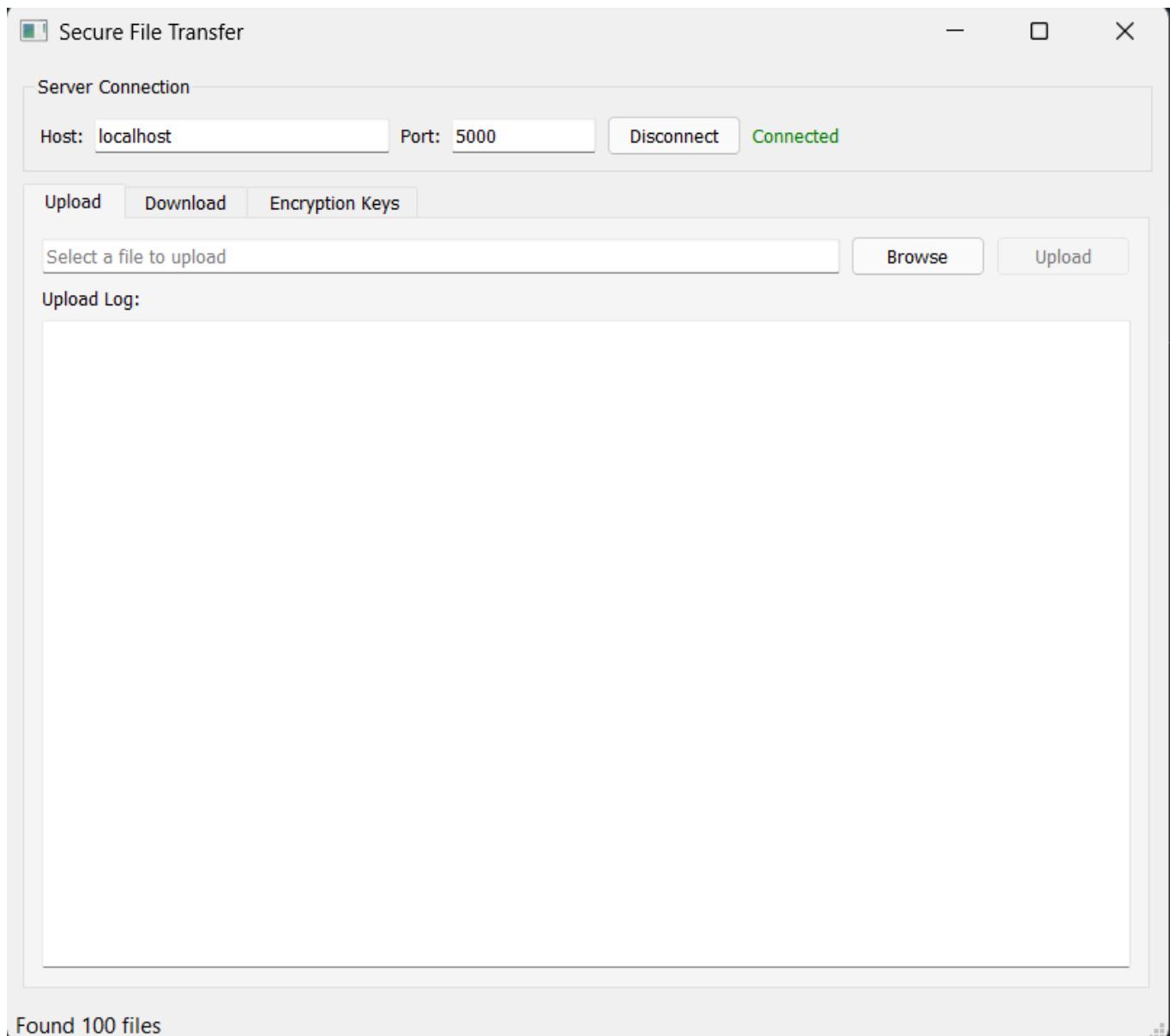
```
PS C:\Github\CN-Project\secure_transfer\Server> python run_server.py
Starting server on 0.0.0.0:5000
Upload directory: uploads
Google Drive integration: Enabled
Server started on 0.0.0.0:5000
```

## Client Setup

1. Ensure you have the `Server.crt` file in the client directory
2. Start the client application:

```
python run_client.py
```

3. In the GUI:
  - Enter the server's IP address
  - Enter the server's port number
  - Click "Connect" to establish a secure connection

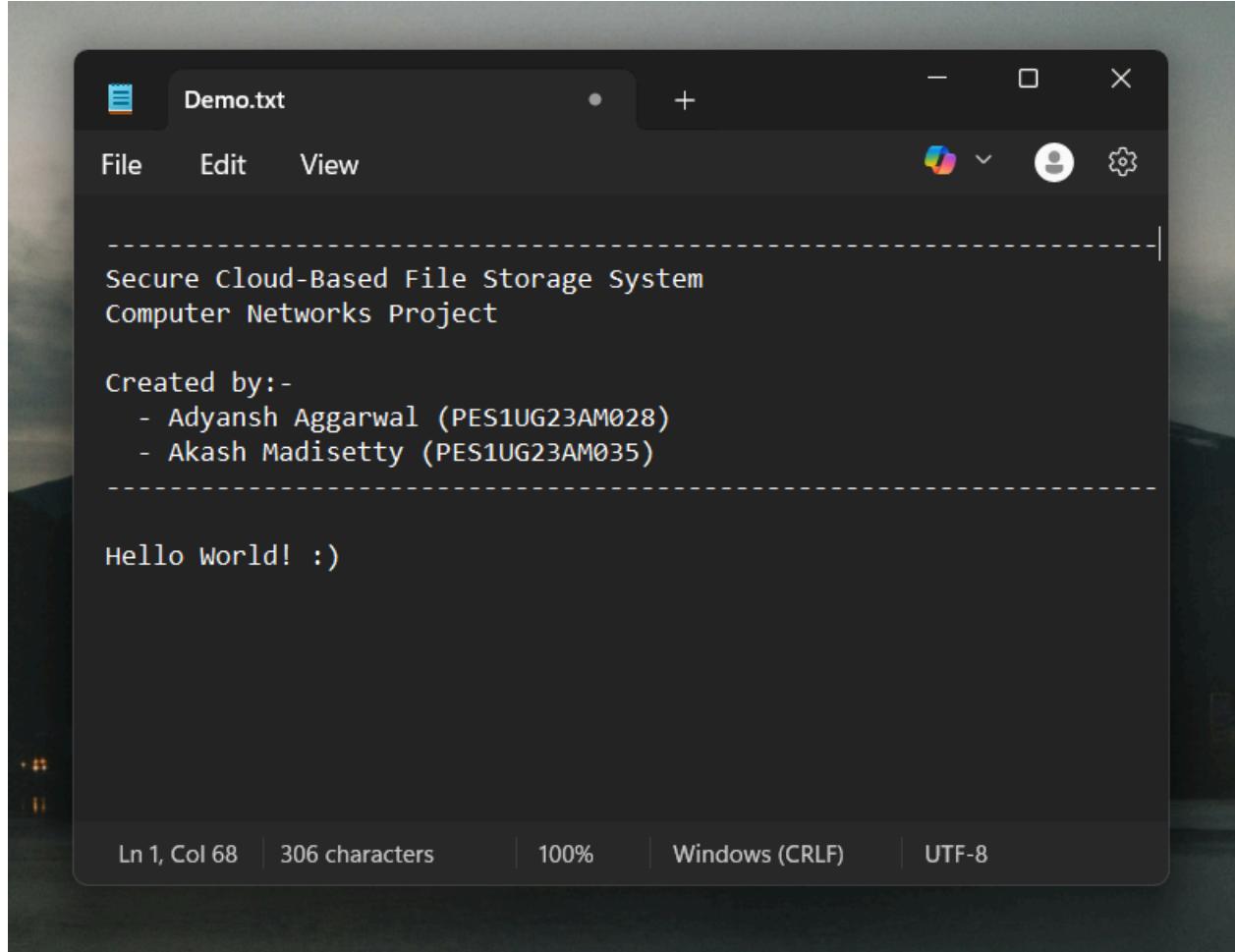


## Using the Application

### 1. File Upload:

- Click "Upload" button
- Select the file to upload
- Choose or generate an AES key (clicking upload automatically generates a new key make sure to save this)
- The file will be encrypted and uploaded to Google Drive

- Let us upload a `demo.txt` for this example



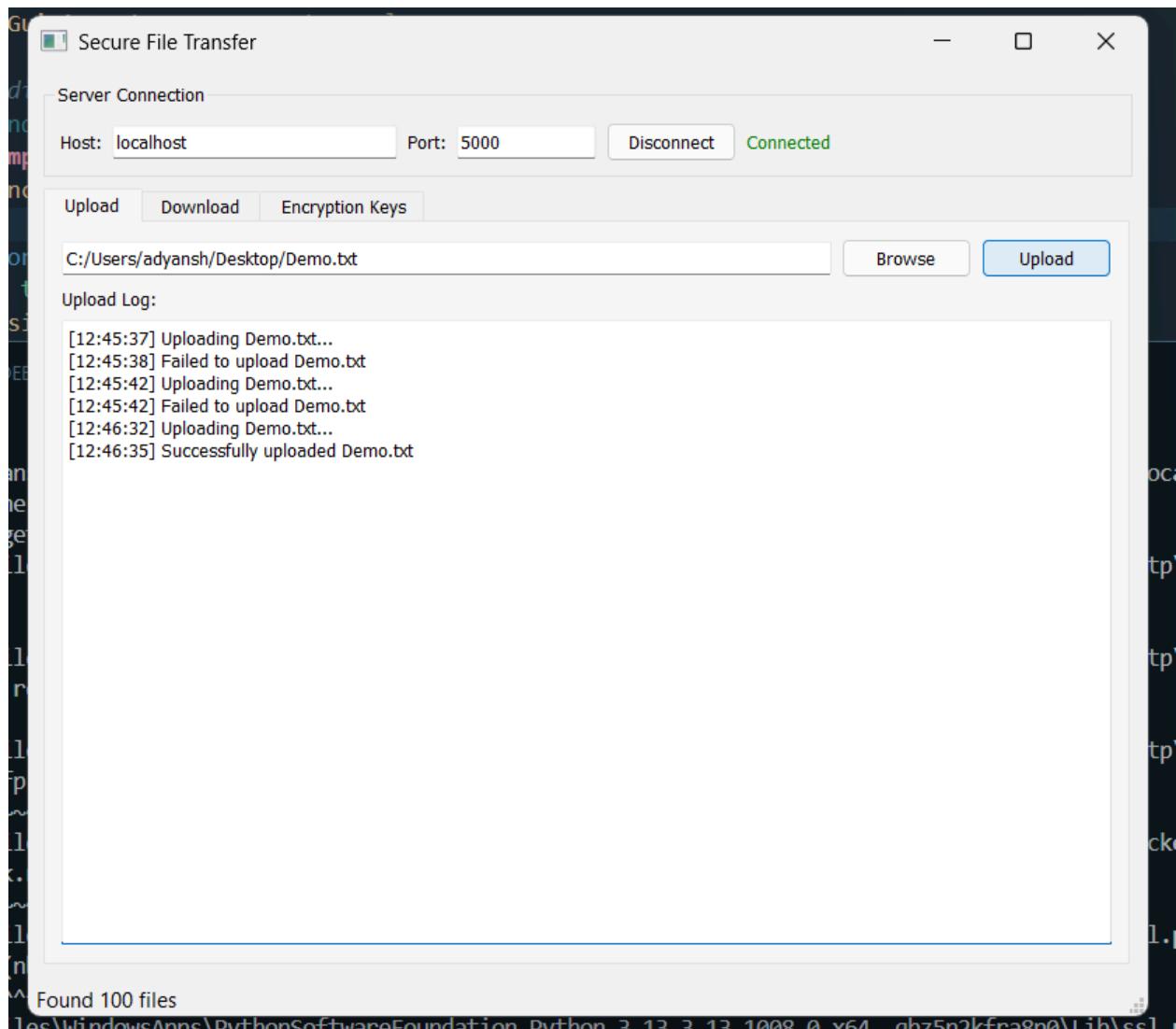
The screenshot shows a dark-themed text editor window titled "Demo.txt". The content of the file is as follows:

```
Secure Cloud-Based File Storage System
Computer Networks Project

Created by:-
- Adyansh Aggarwal (PES1UG23AM028)
- Akash Madisetty (PES1UG23AM035)

Hello World! :)
```

At the bottom of the editor, there are status indicators: "Ln 1, Col 68", "306 characters", "100%", "Windows (CRLF)", and "UTF-8".



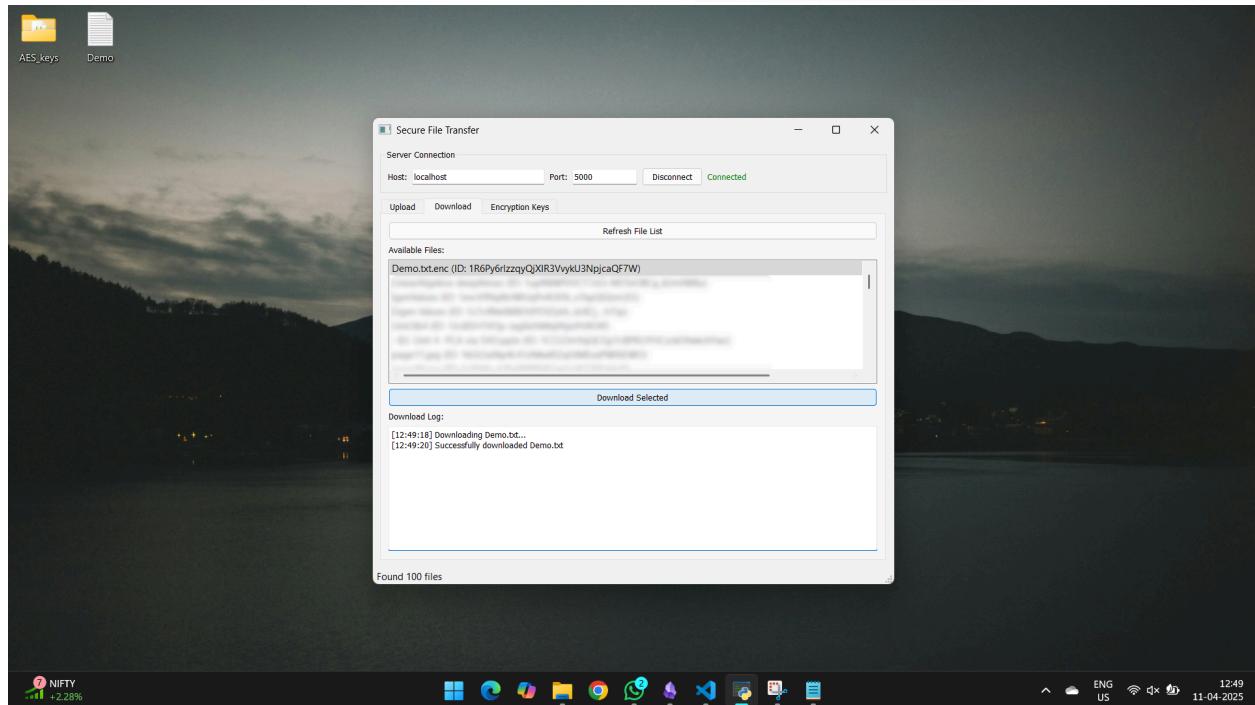
Found 100 files

**#Note :** The Server was disconnected to demonstrate the error logs when file fails to upload

## 2. File Download:

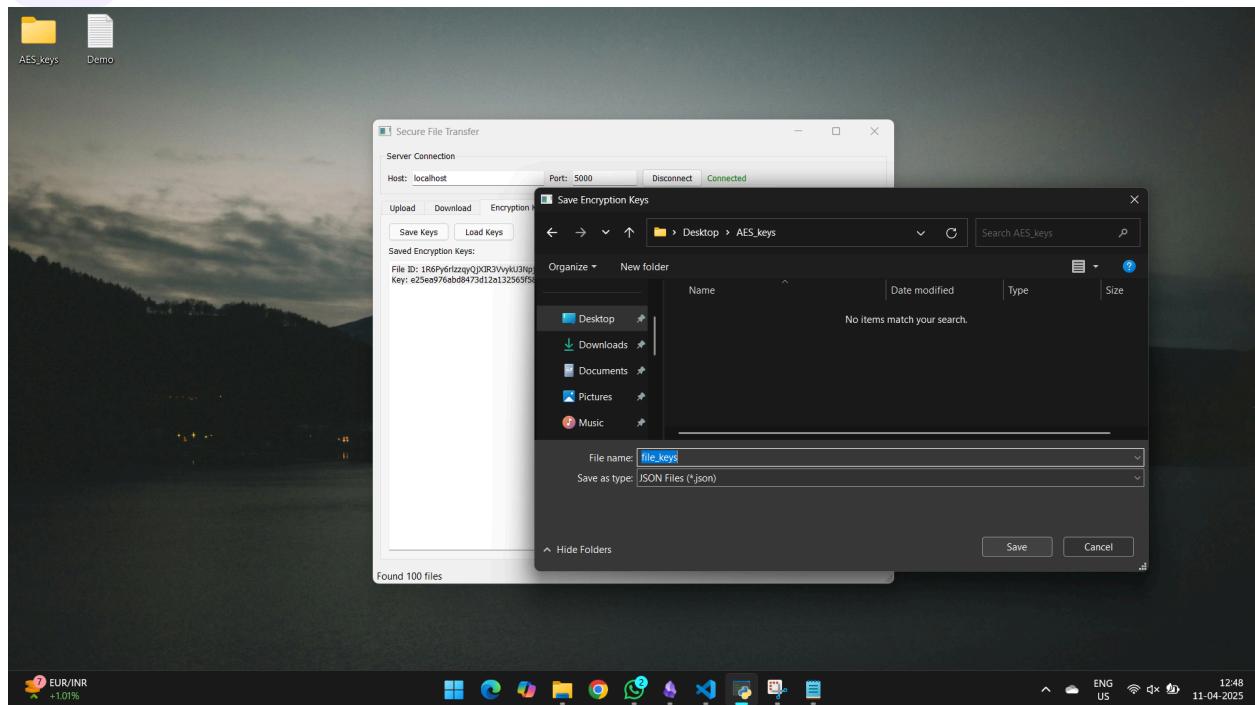
- Select the file from the list
- Provide the correct AES key
- Click "Download"

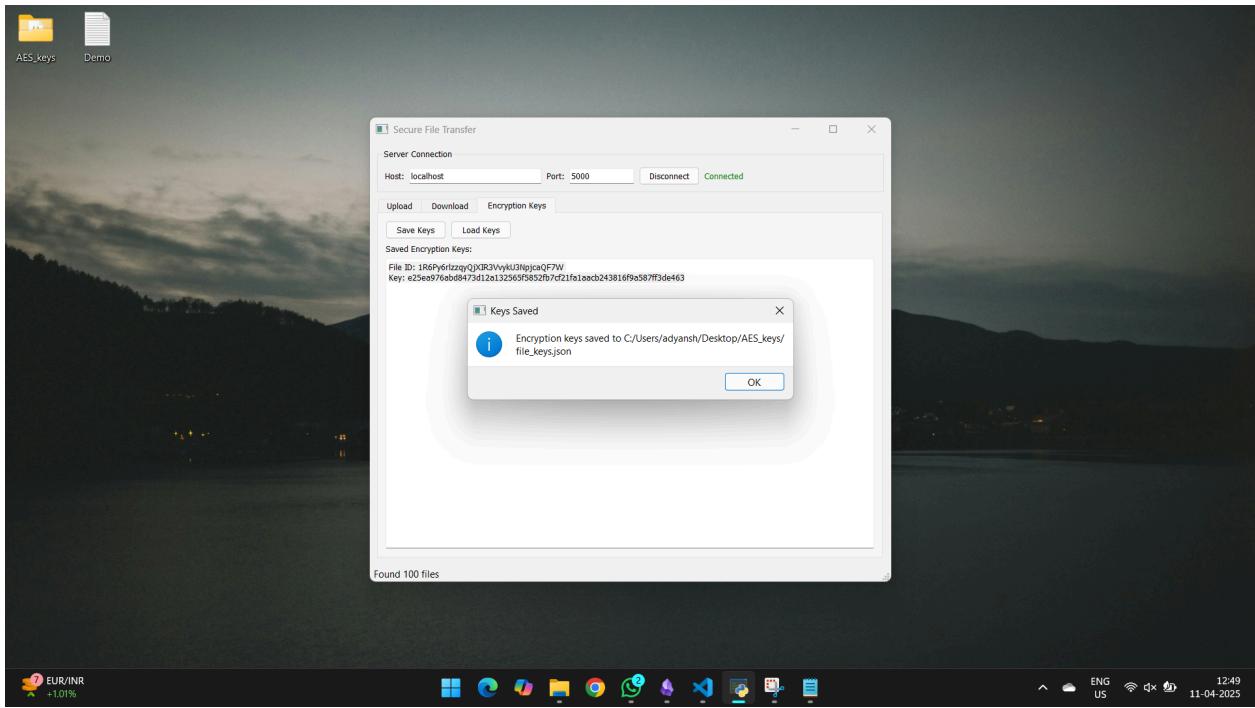
- The file will be decrypted and is saved locally in /client/downloads



### 3. Key Management:

- Load a new AES key and when used for a file make sure to save it.
- Store keys securely as they are required for decryption
- #Note :** Lost keys cannot be recovered





## 4. Results

- The uploaded file will be stored with a `.enc` extension in the Drive. This cannot be viewed even in the drive as the file extension is not supported.

- Incase this file is downloaded from the Drive or a wrong AES key is used the file that is downloaded is corrupted (original `.enc` file is unchanged in the Drive)
- Corrupted file -

```

1 9k:w`dwdRxC4,xCExFBx84t/x80f>x9FxX5xD3%xE0xExE7x90/L6sxD92^xB2xCAM*
2 xDAXE2.2oXFdxE3xCACKxEBx8DmxA2cyx84xD7xA2x8DxFEQ
3 x8Dx8D2x99xA9DxC9n 9",TxF7xBAxFxDEO*x96cwxE5MR-x87D5
4 xDC4x90x82x8A5xD6xF9xA4xFBxEBx83xC1!xC0mpx86xCDkAxFFxB6"xA6xB3x9Dx8Fx

```

## Edge Cases

- Wrong encryption key provided/ no encryption key provided - in such a case the file still downloads but the data is corrupted.

Demo.txt

```
client > downloads > Demo.txt
1  9k: "dwDR", "t/"f>?0%/ENQ Ls2^?RSM]NV?ETB C?)?38?Y?i=?son?SYN?o)?a?/?LJD1 VT?
2  ??bs .2o??So ?KC?FF ??cy ??SOHSUB ??BEL?Q
3  ??2??Dn 9", Tack ??O*?GS CW?MR-?ENDCD5
4  ??4??C?VT ??SYN?EN ??STX?ETX? !FF ??mp ??kA???" ??S?CS ??FF ??W? ??TH? ??RS ??p?d?D?NUL?}Y??Z?ACK?VT?h?7?esc?GS ?? ?? ??NAKDEL?
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Saved encryption keys to C:/Github/CN-Project/secure_transfer/client/file_keys.json
Loaded encryption keys from C:/Github/CN-Project/secure_transfer/server/file_keys.json
Warning: No encryption key found for this file
Downloading daniel-leone-v7daTKlzzaw-unsplash.jpg...
Warning: No encryption key found. File remains encrypted: downloads\daniel-leone-v7daTKlzzaw-unsplash.jpg
Warning: No encryption key found for this file
Downloading Demo.txt...
Warning: No encryption key found. File remains encrypted: downloads\Demo.txt
```

- SSL/TLS certificate or key files are missing or incorrect - client fails to launch

```
PS C:\Github\CN-Project\secure_transfer\Client> python run_client.py
Traceback (most recent call last):
  File "C:\Github\CN-Project\secure_transfer\Client\run_client.py", line 13, in <module>
    main()
    ^^^^
  File "C:\Github\CN-Project\secure_transfer\Client\run_client.py", line 8, in main
    window = MainWindow()
  File "C:\Github\CN-Project\secure_transfer\Client\gui.py", line 74, in __init__
    self.client = FileClient()
    ^^^^^^
  File "C:\Github\CN-Project\secure_transfer\Client\client.py", line 25, in __init__
    self.ssl_context.load_verify_locations('server.crt')
    ^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory
```

- Missing or invalid `credentials.json`

```
PS C:\Github\CN-Project\secure_transfer\Server> python run_server.py
Starting server on 0.0.0.0:5000
Upload directory: uploads
Google Drive integration: Enabled
Failed to initialize Google Drive API: [Errno 2] No such file or directory: 'credentials.json'
Server started on 0.0.0.0:5000
```

```
PS C:\Github\CN-Project\secure_transfer\client> python run_client.py
Connected to server at localhost:5000
Failed to list files: Google Drive integration not enabled
```

## Troubleshooting

### 1. Connection Issues:

- Verify server IP and port
- Check if server is running
- Ensure SSL certificate is valid

### 2. Authentication Errors:

- Verify Google Drive credentials
- Check SSL certificate validity
- Ensure proper key management

### 3. File Transfer Issues:

- Check network connectivity (Client and Server must be on the same network)
- Verify file permissions
- Ensure sufficient storage space