IBM Innovation Center for Education

PHEMESOFT Software That Matters

TRUSTED DELIVERY PARTNER for IBM ICE

University of Petroleum and Energy Studies

Low Level Design
on
Cloud Based Secure Text Transfer

BACHELOR OF TECHNOLOGY

Specialization in Cloud Computing and Virtualization Technology

Team members:

- Anant Tayal (R110219011)
- Ayush Kumar (R110219033)
- Amresh Garg (R110219009)
- Ashutosh Uniyal (R110219026)
- Mayank Agrawal (R11021082)

Guided By-

- Surbhi Saraswat

Industry Mentor-

- Yogesh Ghorpade

Table of Content

Scope of the document

One of the primary issues in the field of cloud computing is cloud security. Personal and sensitive information should never be kept on a third-party storage media since doing so greatly increases the danger of data theft and exploitation by individuals with harmful intentions. Governments and many other large companies have been discouraged from moving their operations to a cloud platform because the threat is so enormous. In the context of the cloud, the conventional techniques for protecting files and information are unnecessary. To make the cloud more dependable and safer, much research and study are being conducted in this area.

Some of the techniques that stand out in this massive body of research include Diffie Hellman Key Exchange and AES encryption.The latter approach is so strong that even the most advanced computers today may require millions of years to decipher the code and read the file. Our solution suggests encrypting the file using any accepted encryption technology and employing Diffie Hellman user authentication. In this manner, the data may be safely kept in a public location without fear of illegal access.

Intended audience

The main aim of our project is to provide secure storage of the encrypted files to the working professional and to the students.

System overview

The fundamental goal of this project was to offer a cloud-based file storage solution that was as safe as feasible. The assault man in the middle forced us to abandon our first plan of online text encryption. About an attack known as NFS, we also learned that. NFS in an attack that may compute a key of order 2768. This included around 232 digits. As a result, we deduced that a bigger prime number is required for the operation. As a result, we employed a prime number of 600 digits. We used a programme to encrypt the file directly on the owner's PC to give them more control.

This project works on the two different parts

1. Graphical User Interface
2. Web Application

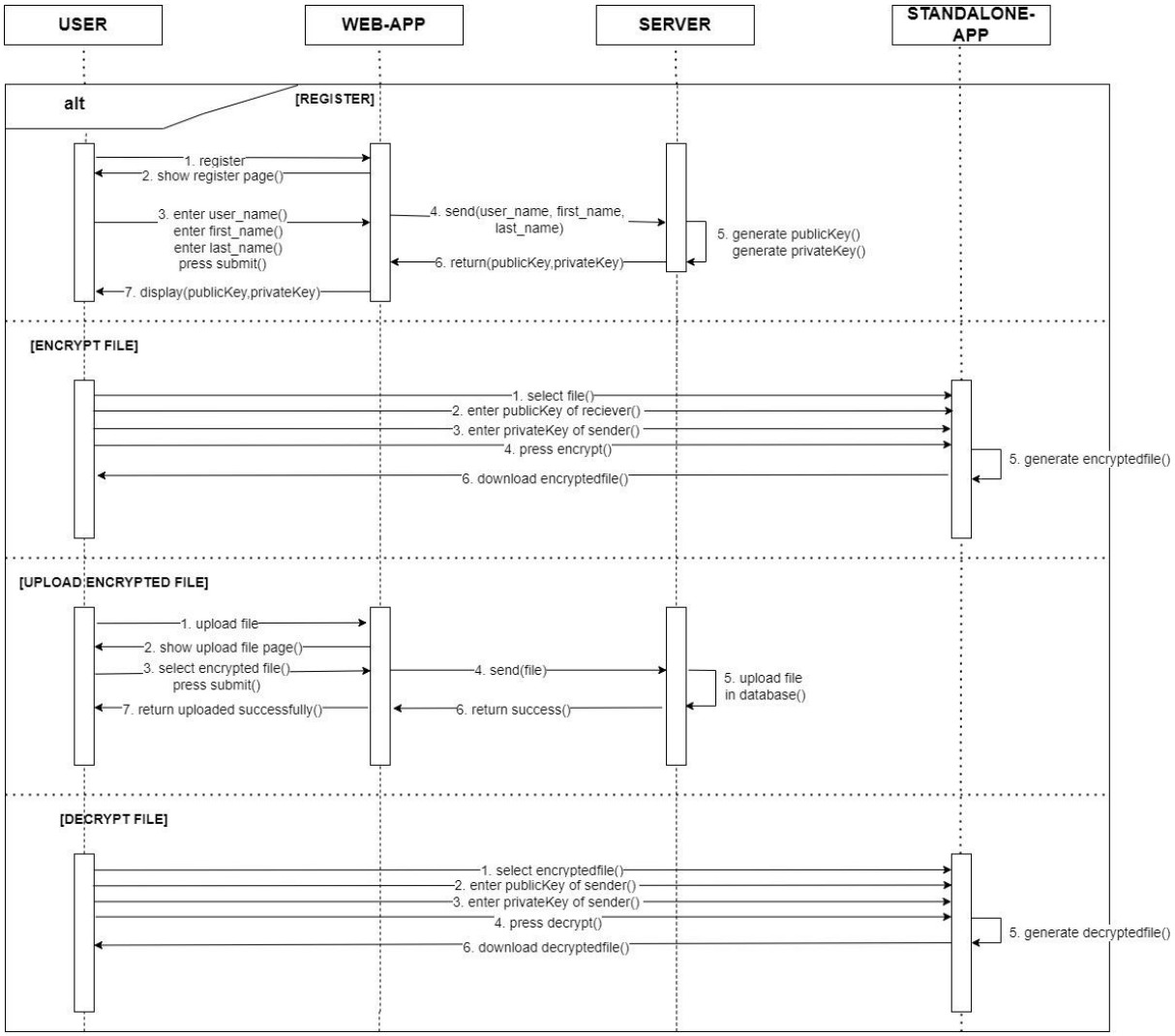
Sequence Diagram

In the given sequence diagram, the user object has four options to select from. The user can register himself/herself, encrypt the file, upload the encrypted file and decrypt the file. If the user selects the register option, a request will be sent to the web app and the web app will return the register page in response. Then the user fills in all the required details like username, firstname, and lastname and then presses the submit button. These details will be sent to the server and the server will generate the public key for the user and send it back to the web app. The web app will then display the public key generated and the user download that public key.

If the user selects the encrypted file option, then a standalone app will be launched. Then the user selects the file which needs to be encrypted from his/her directory and enters other details like the public key of the receiver and the private key of the sender. Then he/she can download the encrypted file by clicking on the download button.

If the user selects upload the encrypted file option, then a request is sent to the web app and it returns the file upload page in response. Then the user can select the encrypted file from his/her directory and click on submit button. Then the web app sends the file to the server where the server saves that file in its database and then the server returns a success message to the web app. The web app shows a file upload successfully message to the user.

If the user selects the decrypt file option, then a standalone app will be launched. Then the user selects the encrypted file from his/her local directory and enters other details like the public key of the user and the private key of the sender. Then he/she can download the decrypted file by clicking on the download button.


USER
WEB-APP
SERVER
STANDALONE-APP
alt
[REGISTER]
1. register
2. show register page()
3. enter user_name()
enter first_name()
enter last_name()
press submit()
4. send(user_name, first_name, last_name)
5. generate publicKey()
generate privateKey()
6. return(publicKey,privateKey)
7. display(publicKey,privateKey)
[ENCRYPT FILE]
1. select file()
2. enter publicKey of reciever()
3. enter privateKey of sender()
4. press encrypt()
5. generate encryptedfile()
6. download encryptedfile()
[UPLOAD ENCRYPTED FILE]
1. upload file
2. show upload file page()
3. select encrypted file()
press submit()
4. send(file)
5. upload file
in database()
6. return success()
7. return uploaded successfully()
[DECRYPT FILE]
1. select encryptedfile()
2. enter publicKey of sender()
3. enter privateKey of sender()
4. press decrypt()
5. generate decryptedfile()
6. download decryptedfile()

Activity Diagram

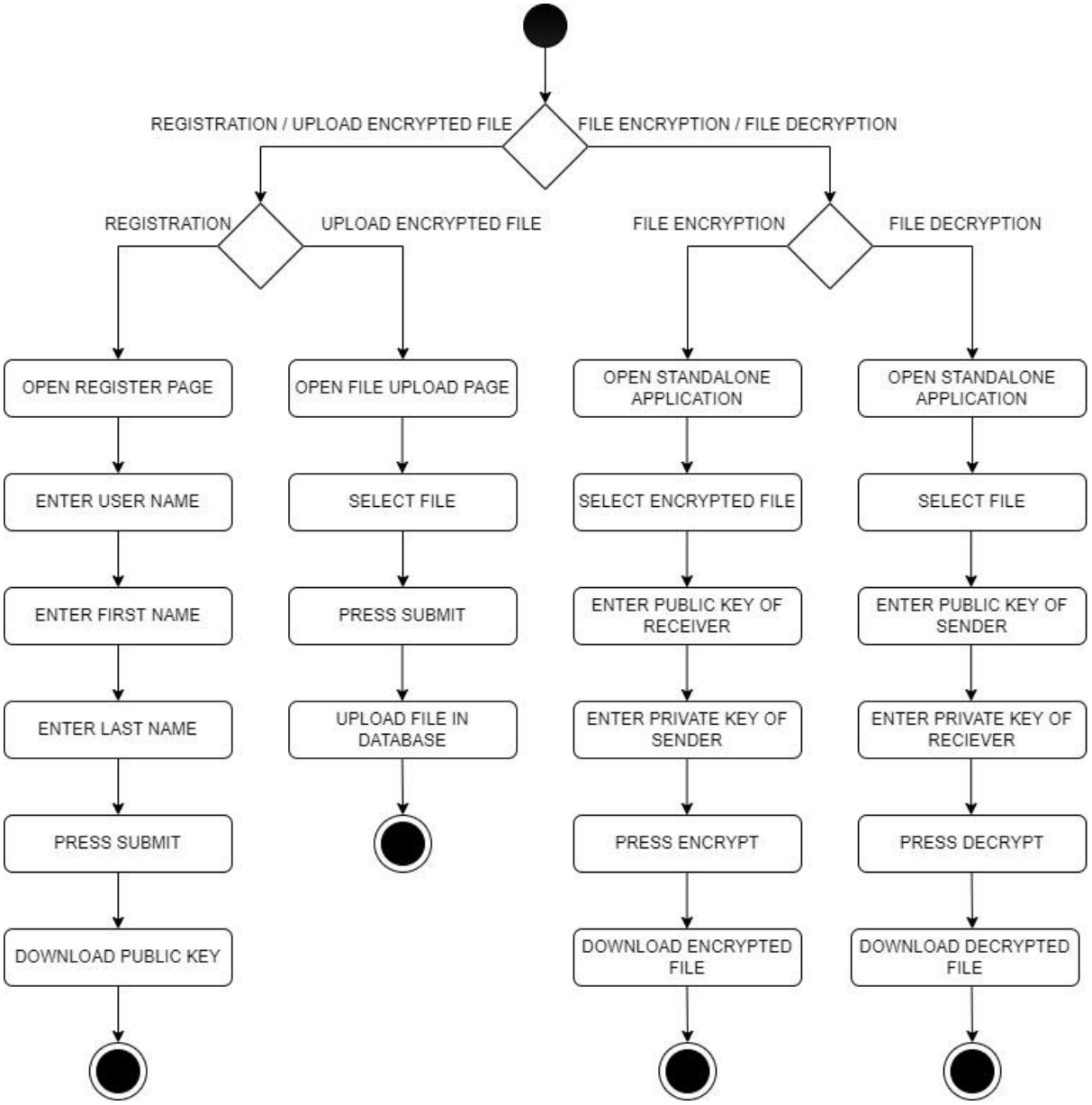
In the given activity diagram, the user is provided with two options i.e., REGISTRATION/UPLOAD ENCRYPTED FILE AND FILE ENCRYPTION/FILE DECRYPTION. If the user selects the first option, then again two choices are there i.e., either REGISTRATION or UPLOAD ENCRYPTED FILE. If the user selects the second option, then again two choices are there i.e. either FILE ENCRYPTION or FILE DECRYPTION.

If the user proceeds with the REGISTRATION choice, then a registration web page will be opened up. The user has to fill in the details including user name, first name, and last name. Then he/she clicks on submit button. Then, the user will receive a public key generated by the server which he/she can download.

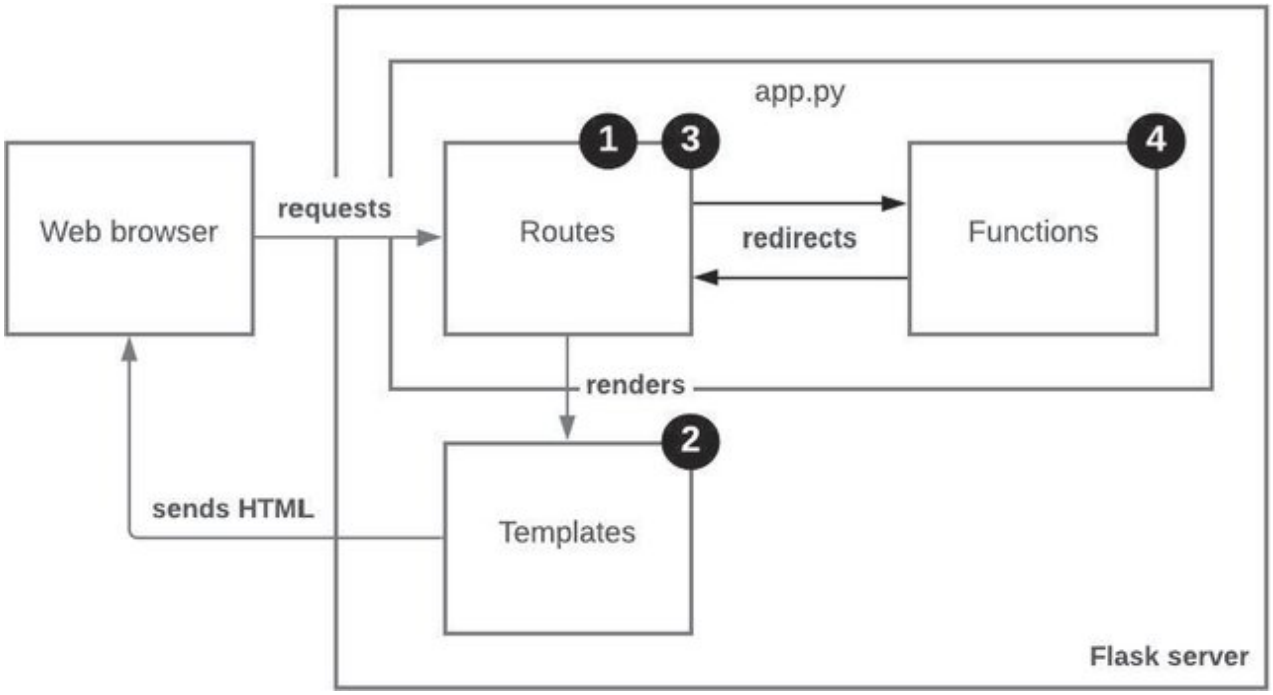
If the user proceeds with the UPLOAD ENCRYPTED FILE choice, a file upload web page will be opened up. The user selects the encrypted file from his/her local directory and clicks on the upload button. The file gets saved in the server's database.

If the user proceeds with the FILE ENCRYPTION choice, then a standalone application will be launched where the user selects the file from his/her local directory and enters other details including the public key of the receiver and private key of the user. The user then clicks on the encrypt button and the encrypted file gets downloaded.

If the user proceeds with the FILE DECRYPTION choice, then a standalone application will be launched where the user selects the encrypted file from his/her local directory and enters other details including the public key of the sender and the private key of the user. The user then clicks on the decrypt button and the decrypted file gets downloaded.


REGISTRATION / UPLOAD ENCRYPTED FILE
FILE ENCRYPTION / FILE DECRYPTION
REGISTRATION
UPLOAD ENCRYPTED FILE
FILE ENCRYPTION
FILE DECRYPTION
OPEN REGISTER PAGE
ENTER USER NAME
ENTER FIRST NAME
ENTER LAST NAME
PRESS SUBMIT
DOWNLOAD PUBLIC KEY
OPEN FILE UPLOAD PAGE
SELECT FILE
PRESS SUBMIT
UPLOAD FILE IN DATABASE
OPEN STANDALONE APPLICATION
SELECT ENCRYPTED FILE
ENTER PUBLIC KEY OF RECEIVER
ENTER PRIVATE KEY OF SENDER
PRESS ENCRYPT
DOWNLOAD ENCRYPTED FILE
OPEN STANDALONE APPLICATION
SELECT FILE
ENTER PUBLIC KEY OF SENDER
ENTER PRIVATE KEY OF RECIEVER
PRESS DECRYPT
DOWNLOAD DECRYPTED FILE

Component Diagram



Details of framework

Python-based Flask is a microweb framework. Since it doesn't require any specific tools or libraries, it is categorized as a microframework. [2] It lacks any components where pre-existing third-party libraries already provide common functionality, such as a database abstraction layer, form validation, or other components. However, Flask allows for extensions that may be used to add application functionalities just as they were built into the core of Flask. There are extensions for object-relational mappers, form validation, upload handling, several open authentication protocols, and several utilities associated with popular frameworks.

Session Management

Flask-Session is an extension for Flask that supports Server-side Session to your application. The Session is the time between the client logs in to the server and logs out of the server. The data that is required to be saved in the Session is stored in a temporary directory on the server. The data in the Session is stored on the top of cookies and signed by the server cryptographically. Each client will have their own session where their own data will be stored in their session.

Caching

Flask-Caching is an extension to Flask that adds caching support for various backends to any Flask application. By running on top of cachelib it supports all of werkzeug's original caching backends through a uniformed API. It is also possible to develop your caching backend by sub classing the 'flask_caching.backends.base.BaseCache' class.

Unit Testing

Unit tests test the functionality of an individual unit of code isolated from its dependencies. They are the first line of defense against errors and inconsistencies in your codebase. They test from the inside out, from the programmer's point of view.

In Python, we can use the-

1. pytest is a test framework for Python used to write, organize, and run test cases. After setting up your basic test structure, pytest makes it easy to write tests and provides a lot of flexibility for running the tests
2. Python has a built-in test framework called unit test, which is a great choice for testing as well. The unit test module is inspired by the xUnit test framework.

Flask unit testing is defined as a method of testing individual portions of source code that comprises one or more modules programmed together, which takes care of procedures of usage, procedures of operation, and the data associated with it to be tested to determine their fitness for usage. These tests are run by software developers so that one is ensured that the section of the code written for a specific task accomplishes the same along with meeting the design requirements and the behavior. These tests are typically automated tests targeted at individual functions or procedures.

The process of performing the unit tests in Flask

1. Storing the unit tests
2. Creation of the basic unit test file
3. Running of unit tests
4. Addition of more helper functions for facilitating more unit test development

Reference

- Diffie, W.; Hellman, M. (1976). "New directions in cryptography" (PDF). IEEE Transactions on Information Theory. 22 (6): 644–654. doi:10.1109/TIT.1976.1055638. Archived (PDF) from the original on 2014-11-29.
- Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Archived from the original on 23 June 2012.
- "About Python". Python Software Foundation. Retrieved 24 April 2012., second section "Fans of Python use the phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files."
- https://testdriven.io/blog/flask-pytest/
- https://www.educba.com/flask-unit-testing/
- https://www.geeksforgeeks.org/how-to-use-flask-session-in-python-flask/