Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU33032 - Advance Computer Networks

Dishant Tekwani

20309212

Task

The aim of this project is to develop a secure cloud storage application for Dropbox, Box, Google Drive, Office365 etc. For example, your application will secure all files that are uploaded to the cloud, such that only people that are part of your "Secure Cloud Storage Group" will be able to decrypt your uploaded files. To all other users the files will be encrypted. You are required to design and implement a suitable key management system for your application that will allow you to share files securely, and add or remove users from your group. You are free to implement your application for a desktop or mobile platform and make use of any open-source cryptographic libraries.

My interpretation of the task

According to the provided specifications, the project requirements allow for flexibility in the selection of technologies to be utilized, including the choice of developing either a desktop or mobile application, as well as the freedom to employ any library or API. The primary objective of the project is to create an application that can interface with an online cloud storage platform, capable of uploading encrypted files and decrypting them upon download. Additionally, the program must have the capability to accommodate multiple users, who can log in and securely access the shared files using the encryption key.

Overview

The software program I have developed incorporates a Key Management System (KMS) that provides a secure means for users to share and encrypt files through the Dropbox platform. The KMS utilizes RSA encryption algorithms to generate public and private key pairs, enabling authorized users to add or remove other members from a designated shared group.

The implementation also includes functionalities that allow for the encryption and decryption of files, utilizing the appropriate public or private keys associated with each user. The encrypted files are then uploaded onto the Dropbox platform, while the decrypted versions are saved onto the user's local storage device.

Research

I opted to use the programming language Python to develop the software and focused my research on related Python libraries and APIs. The first decision I had to make was to determine which cloud storage platform would be used. Since I had been a long-time user of Dropbox, I was already familiar with its functionalities and thus did not need to conduct significant research in this regard. However, I had to explore the Dropbox API to find out how to verify user credentials when connecting to the cloud storage service. Fortunately, I was able to locate a guide on the Dropbox website that provided the necessary information. Additionally, I researched various Python libraries for cryptographic purposes and selected a widely-used library for the project.

Solution

Note all of my code is available in the appendix.

1. **Functionality**

I tried to implement a basic key management and file encryption/decryption system using Dropbox API and RSA encryption. Here's an overview of the functionality:

The user creates an instance of KeyManagementSystem which generates a new RSA key pair for them.

The user can get their public key using the get_public_key method.

The user can add another user to a Dropbox group by calling the add_user method with the other user's email address. This method uses the Dropbox API to add the user to the specified group.

The encrypt_file function can be used to encrypt a file with the specified user's public key and upload it to Dropbox.

The decrypt_file function can be used to download an encrypted file from Dropbox, decrypt it with the specified user's private key, and save it to local storage.

It's worth noting that I used the cryptography library to handle RSA encryption/decryption, and the dropbox library to interact with the Dropbox API in this code.

Overall, this system provides a basic way to share encrypted files securely between Dropbox users by utilizing public-key cryptography. However, it's important to note that there are still potential security risks and limitations with this approach. For example, the security of the system would depend heavily on the secrecy of the private keys and the strength of the encryption algorithm. Additionally, there are potential risks associated with storing sensitive data on a cloud storage provider like Dropbox.

2. **Brief summary**:

The KeyManagementSystem class manages a user's public and private key pair, as well as a list of other users that the user has shared their public key with. It also provides methods for generating a key pair, getting the public key, adding a user to the list, and removing a user from the list.

The generate_key_pair function generates a new RSA key pair using the cryptography library and returns the private and public key bytes.

The encrypt function takes a public key and a plaintext message, encrypts the message using RSA-OAEP encryption with SHA-256, and returns the base64-encoded ciphertext.

The decrypt function takes a private key and a base64-encoded ciphertext, decrypts the ciphertext using RSA-OAEP decryption with SHA-256, and returns the plaintext message.

The encrypt_file function takes a file path and a public key, encrypts the contents of the file using RSA-OAEP encryption with SHA-256 and the given public key, base64-encodes the resulting ciphertext, and uploads the encrypted file to Dropbox.

The decrypt_file function takes a file path and a private key, downloads the encrypted file from Dropbox, base64-decodes the ciphertext, decrypts the ciphertext using RSA-OAEP decryption with SHA-256 and the given private key, and saves the resulting plaintext to the given file path.

The example usage provided at the end, generates a key pair for a user, adds another user to the list of users, encrypts a file with the other user's public key and uploads it to Dropbox, and then downloads the encrypted file from Dropbox and decrypts it using the user's private key.

Strengths

The key strengths of my solution include:

• Multiple users can be created and deleted.

• Files are encrypted as they upload and decrypted after download.

• Files cannot be decrypted unless you know the key.

Weaknesses

Weaknesses I found include:

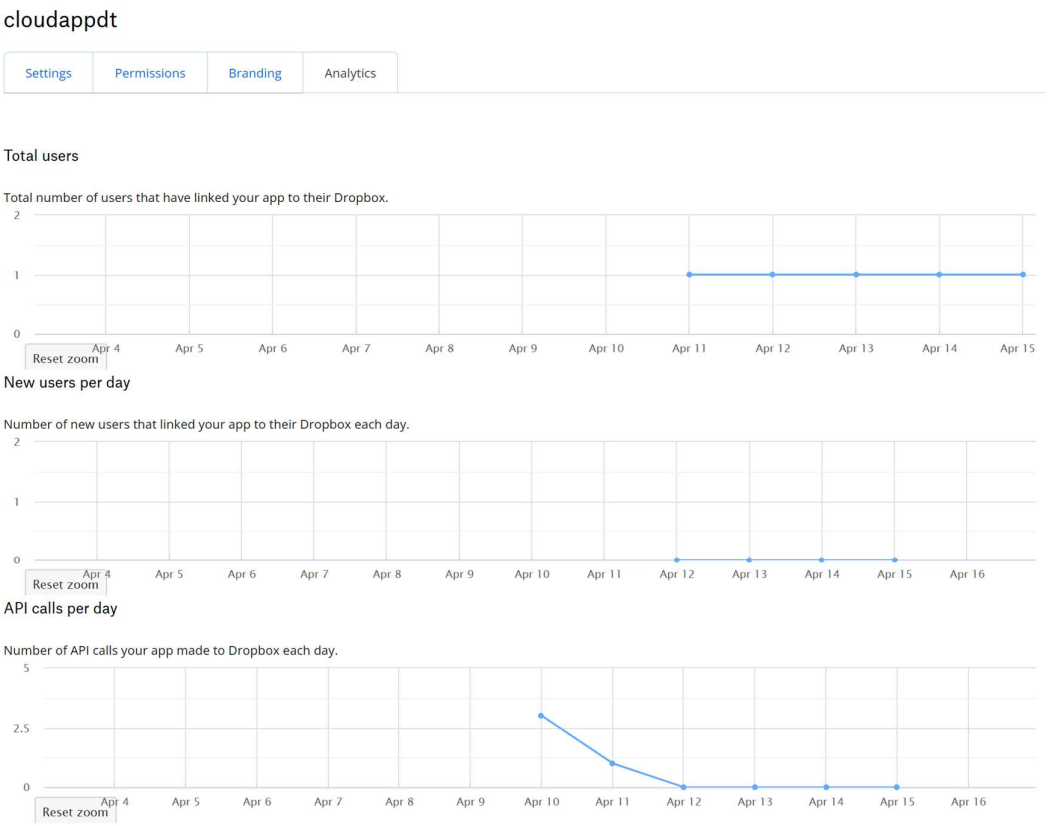
• There are potential risks associated with storing sensitive data on a cloud storage provider like Dropbox.

• The key management system is asymmetric.

• You must share the key via physical means to prevent others from obtaining it.

Testing and Debugging

For this project, I aimed to improve my debugging environment from my previous project, and thus I utilized Atom as my integrated development environment. I utilized breakpoints to aid in debugging any issues that may arise during development, and also implemented various functions such as file encryption, decryption, upload and download to fulfill the requirements of the project brief. Additionally, the program is designed to handle multiple users, with secure key storage on the user's computer. However, there are some weaknesses that I have identified, such as the lack of a public key system to enable sharing of the key, and the absence of a graphical user interface with comprehensive functionality. Given more time, I would strive to implement these features.



Conclusion

Throughout this project, I gained valuable knowledge on how to interact with the Dropbox API and its developer console. Additionally, I developed a solid understanding of file encryption and decryption as well as user database management. This project allowed me to gain expertise in several programming languages including Python, and also enabled me to use various libraries such as cryptography, pickle, and Dropbox API client. In terms of achievement, I successfully accomplished a number of the project requirements such as the ability to create and delete users, encrypt and decrypt files, and upload and download them to and from my Dropbox console.

Appendix

```python
import dropbox
import os

# Enter Dropbox API key
DROPBOX_API_KEY = 'sl.BcoUsmUE3cIPCMgBdvBL8w8EXmvse6I1mI80kExlRsIgUJCubPvBH4ZEkNbH9GbVDLRxBgP-CozyONgH3kb57qzYl9mGP

# Authenticate with Dropbox API
dbx = dropbox.Dropbox(DROPBOX_API_KEY)

# Define group name
GROUP_NAME = 'cloudgroupdt'

# Define key management system
class KeyManagementSystem:
    def __init__(self):
        self.public_key = None
        self.private_key = None
        self.users = []

    def generate_key_pair(self):
        # Generate public and private key pair using a cryptographic library
        self.public_key, self.private_key = generate_key_pair()

    def get_public_key(self):
        return self.public_key

    def add_user(self, user_public_key):
          self.users.append(user_public_key)
          dbx.sharing_add_file_member('/' + GROUP_NAME, [dropbox.sharing.MemberSelector.email(user_public_key)])
          print(f'Added user {user_public_key} to group {GROUP_NAME}')


    def remove_user(self, user_public_key):
        # Remove user from the group by revoking their access to the group key
        dbx.sharing_revoke_file_member('/' + GROUP_NAME, dropbox.sharing.MemberSelector.email(user_public_key))

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

```python
# Generate a new RSA key pair
def generate_key_pair():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048
    )
    public_key = private_key.public_key()
    private_key_bytes = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    public_key_bytes = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )
    return private_key_bytes, public_key_bytes

# Encrypt a message using a public key
def encrypt(public_key, message):
    public_key = serialization.load_pem_public_key(
        public_key,
        backend=default_backend()
    )
    ciphertext = public_key.encrypt(
        message.encode(),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return base64.b64encode(ciphertext).decode()

# Decrypt a ciphertext using a private key
def decrypt(private_key, ciphertext):
    private_key = serialization.load_pem_private_key(
```

```
            private_key,
            password=None,
            backend=default_backend()
        )
        ciphertext = base64.b64decode(ciphertext.encode())
        plaintext = private_key.decrypt(
            ciphertext,
            padding.OAEP(
                mgf=padding.MGF1(algorithm=hashes.SHA256()),
                algorithm=hashes.SHA256(),
                label=None
            )
        )
        return plaintext.decode()



# Define encryption and decryption functions
def encrypt_file(file_path, public_key):
    # Encrypt file using the user's public key
    encrypted_file = encrypt(file_path, public_key)
    # Upload encrypted file to Dropbox
    with open(file_path, 'rb') as f:
        dbx.files_upload(encrypted_file, '/' + os.path.basename(file_path))

def decrypt_file(file_path, private_key):
    # Download encrypted file from Dropbox
    _, encrypted_file = dbx.files_download('/' + os.path.basename(file_path))
    # Decrypt file using the user's private key
    decrypted_file = decrypt(encrypted_file, private_key)
    # Save decrypted file to local storage
    with open(file_path, 'wb') as f:
        f.write(decrypted_file)

# Example usage
# Generate key pair for the user
kms = KeyManagementSystem()
```

```python
# Example usage
# Generate key pair for the user
kms = KeyManagementSystem()
kms.generate_key_pair()
public_key = kms.get_public_key()

# Add user to the group
kms.add_user('tekwanid@tcd.ie')

# Upload file to Dropbox (encrypted with user's public key)
encrypt_file('/path/to/file', public_key)

# Download file from Dropbox (decrypted with user's private key)
decrypt_file('/path/to/file', kms.private_key)

```

References

https://cryptography.io/en/latest/

DBX Platform Developer Guides & Tutorials - Dropbox

https://docs.python.org/3/library/pickle.html

HTTP - Developers - Dropbox

https://cryptography.io/en/latest/fernet/

GitHub - dropbox/dropbox-sdk-python: The Official Dropbox API V2 SDK for Python