CS6301_Report.pdf

by Meet Nirav Zaveri

Submission date: 06-Aug-2024 02:20PM (UTC-0500)
Submission ID: 2372833175
File name: CS6301_Report.pdf (1.17M)
Word count: 2400
Character count: 13980

Securing Cloud Storage

Varshil Rajeshbhai Narola
NetID: vxn220007

Gaurav Sharma
NetID: GXS230001

Hanisha Anil Mohinani
NetID: HXM220089

***Abstract*—Data encryption converts information into a coded format that can only be deciphered by individuals possessing the appropriate private key or password. As one of the most prevalent and powerful data security measures employed by organizations, encryption plays a crucial role in protecting sensitive information. This paper delves into the use of symmetric and asymmetric cryptographic techniques to enhance the security of cloud storage solutions. Additionally, it discusses the implementation of a secure cloud storage application utilizing these encryption methods, focusing on the integration with Google Drive and the development of a robust key management system to facilitate secure file sharing and user management.**

I. Introduction

Data encryption converts information into a coded format that can only be deciphered by individuals possessing the appropriate private key or password. Encryption stands as one of the most prevalent and powerful data security measures employed by businesses to safeguard their digital assets. There are two fundamental approaches to data cryptography: symmetric and asymmetric cryptography, also known as public key cryptography. Symmetric key encryption utilizes the same private key for both the encryption and decryption of messages or files. Although it is faster than asymmetric encryption, it requires the sender to securely share the encryption key with the recipient beforehand.

In contrast, asymmetric encryption employs two distinct keys: a public key, which can be freely distributed, and a private key, which must remain confidential. This method ensures that data can be securely transmitted and only deciphered by the intended recipient. Encryption not only protects data stored on local systems but also secures information transmitted over the Internet and other networks. This paper focuses on implementing these cryptographic methods to enhance the security of cloud storage applications, particularly through the integration with Google Drive and the development of an efficient key management system.

II. Objective

Public key cryptography is widely utilized for end-to-end encryption. Although symmetric encryption can be used under normal circumstances, it has inherent limitations when the message needs to be securely transmitted over a channel with potential eavesdroppers and insecure intermediaries. Encrypting the message before sending it and decrypting it after it reaches the recipient is crucial to ensure that only the intended recipient can understand it. However, symmetric encryption requires both parties to share the same key, which poses a risk if the key is intercepted during transmission.

Symmetric encryption alone cannot sufficiently guarantee the security of the transmission, as an attacker could intercept and steal the symmetric key, perform a man-in-the-middle attack, or alter the message contents. This can result in the recipient receiving a modified or incorrect message. To address these vulnerabilities, an additional layer of encryption is needed to protect the symmetric key. One effective method is to encrypt the symmetric key using the recipient's public key and decrypt it with the corresponding private key.

The objective of this project is to implement a secure cloud storage application that leverages public key cryptography to safeguard the symmetric key during transmission, ensuring that only authorized recipients can decrypt and access the data. This approach also aims to protect the metadata, such as the subject, date, sender, and recipient, from being exposed to intermediaries. By integrating this method with Google Drive, the project aims to provide a robust solution for secure file sharing and key management within a cloud storage environment.

A. Problem Statement

The aim of this project is to create a secure cloud storage application compatible with services such as Dropbox, Box, Google Drive, and Office365. The application must ensure that all data uploaded to the cloud is encrypted, allowing only members of the "Secure Cloud Storage Group" to decrypt and access the files, while keeping the data encrypted for all other users. Additionally, the solution should facilitate the addition and removal of users from the secure group.

While encryption can prevent unauthorized network snoopers from accessing the messages, it is also crucial to protect the file contents from potential attacks. This includes verifying the sender's identity using public key certificates. To achieve this, a comprehensive key management system will be developed and deployed, enabling secure file sharing and dynamic user management within the Secure Cloud Storage Group.

The project will leverage open-source cryptographic libraries that can be installed on both desktop and mobile devices. The application will focus on maintaining data security throughout its lifecycle, ensuring that sensitive information is protected during storage and transmission. By integrating these security measures with Google Drive, the project aims to provide a robust and user-friendly solution for secure cloud storage.

III. System Design

The system design focuses on securely transmitting the symmetric key to authorized members of the cloud storage group. Symmetric keys are utilized for both encryption and decryption during the uploading and downloading of files. Given that the activities of the cloud storage group are continually monitored and visible to potential intruders, it is imperative to encrypt the contents of the cloud storage. Only users with the symmetric key should be able to decrypt the data. To achieve this, the symmetric key must be securely shared among users. Users are prompted to send their public key to the cloud storage group, which will then use these keys to encrypt the symmetric key during transit.

For this project, Google Drive is chosen as the cloud storage solution. Detailed technical descriptions will follow, but the primary design is outlined below. When a team member is removed, appropriate measures must be taken to secure the group. Two strategies can be considered: broadcasting the new symmetric key to all users or sending the new symmetric key upon request. The latter method is chosen to reduce algorithm complexity, avoid unnecessary broadcasts, and ensure a new symmetric key is only generated when a user performs an upload or download operation.

A. System Diagrams

The following diagrams illustrate the key components and processes of the system design:

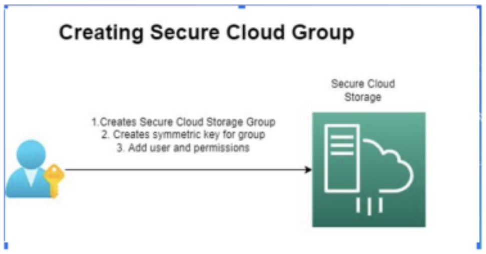

Creating Secure Cloud Group
Secure Cloud Storage
1.Creates Secure Cloud Storage Group
2. Creates symmetric key for group
3. Add user and permissions

Fig. 1. Creating a Secure Cloud Group

Figure 1 depicts the creation of a secure cloud group on the cloud storage platform. This involves generating a symmetric key for the group and adding users with their respective permissions.

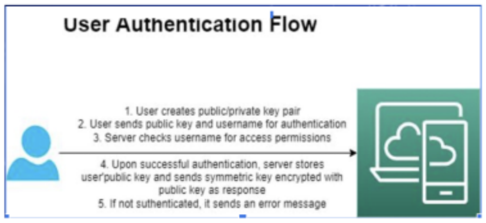

User Authentication Flow
1. User creates public/private key pair
2. User sends public key and username for authentication
3. Server checks username for access permissions
4. Upon successful authentication, server stores user'public key and sends symmetric key encrypted with public key as response
5. If not suthenticated, it sends an error message

Fig. 2. User Authentication Flow

Figure 2 illustrates the user authentication process. Users generate a public and private key pair, send their email and public key for authentication, and the server checks permissions. If authenticated, the server stores the public key and provides the symmetric key encrypted with the user's public key. If authentication fails, an error is returned.

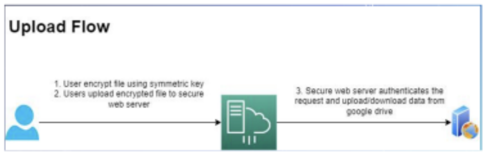

Upload Flow
1. User encrypt file using symmetric key
2. Users upload encrypted file to secure web server
3. Secure web server authenticates the request and upload/download data from google drive

Fig. 3. File Upload Flow

Figure 3 shows the file upload process. The file is encrypted using the symmetric key, and users upload it to the secure cloud storage. The web server authenticates the user before allowing upload or download operations.

B. Secure Key Distribution

In this design, securely distributing the symmetric key is crucial. When a new user joins the group, they send their public key to the cloud storage system, which uses it to encrypt the symmetric key. This ensures that only the intended recipient can decrypt the symmetric key using their private key. The use of public key cryptography for this exchange mitigates the risk of key interception.

C. Encryption and Decryption Process

The system employs symmetric encryption for file storage and asymmetric encryption for key exchange. This combination leverages the speed of symmetric encryption and the security of asymmetric encryption. The encryption process involves the user encrypting files with the symmetric key, which is then securely shared using the recipient's public key.

This design ensures robust data security for cloud storage, protecting against unauthorized access and potential attacks while maintaining efficient and user-friendly operations.

IV. Implementation

The Cloud Storage Group plays a crucial role in overseeing and managing cloud storage operations. Its responsibilities include managing user memberships, re-encrypting files when users are removed, and processing requests from both members and non-members. The system also handles response messaging based on these requests.

Upon the initial startup of the application, it checks for the presence of an encryption key. If no key is found, the system generates a new one. Subsequently, the application verifies whether the user is already registered. Unregistered users are prompted to create an administrator account, which grants the ability to manage user accounts and generate new keys. Existing users are prompted to log in, with administrators receiving access to an extended feature set and regular users seeing a simplified menu. The application operates continuously, allowing users to join or leave the system as needed.

During operation, the application listens for incoming requests through an established connection. These requests include a list of usernames, the listener address of the requesting user, and their public key. The system uses this information to validate user membership. If a request is made by a non-member, an access error is issued. For authenticated members, the system provides the symmetric key encrypted with the member's public key.

When a member is removed from the group, the following actions are undertaken:

- Re-encrypt all files stored in the associated cloud drive with a new symmetric key.
- Generate and securely store a new symmetric key.
- Apply the new symmetric key to re-encrypt all files within the cloud drive.

These measures are implemented to ensure that removed members cannot access or decrypt the contents of the cloud storage. A shared lock mechanism is employed to prevent unauthorized transmissions and substitutions of the symmetric key within the group.

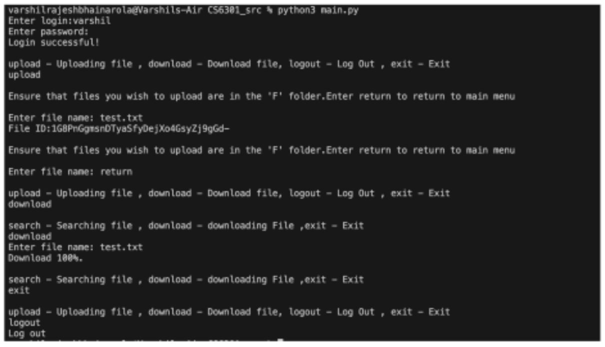
A. Core Files of the System

- **Users.py:** Handles the creation, deletion, and management of usernames and passwords.
- **GoogleDriveAPI.py:** Serves as the intermediary between the system and Google Drive, facilitating file retrieval and uploads.
- **MenuTypes.py:** Manages the creation of menus for both administrative (privileged) and standard (non-privileged) users, invoking other classes to execute required functions.
- **Encryption.py:** Responsible for performing encryption and decryption operations, as well as generating cryptographic keys.
- **Auth.py:** Manages authentication for Google Drive users when interacting with the APIs.

Programming Language: Python
Asymmetric Cryptography: Fernet (AES in CBC Mode with 128-bit key)
Symmetric Cryptography: SHA-256
Additional Considerations:

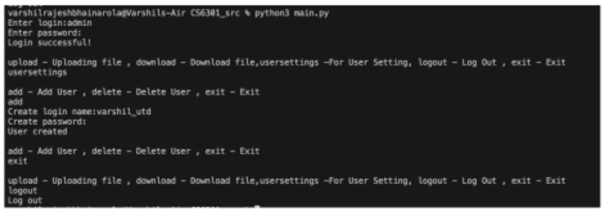
- **Key Management:** Efficient management of cryptographic keys is essential to ensure secure data encryption and decryption, involving key generation, secure storage, and distribution practices.
- **Scalability:** The system should be designed to scale effectively as the number of users and files increases, ensuring sustained performance and security.
- **Security Enhancements:** Regular updates and security patches are necessary to address emerging threats and vulnerabilities, maintaining the robustness of the cryptographic protocols used.
- **User Experience:** A well-designed user interface is critical for simplifying the management of cloud storage operations and user permissions, thereby improving both administrative efficiency and user satisfaction.
- **Compliance:** Ensure that the implementation complies with relevant data protection regulations and standards to guarantee the legality and ethical handling of user data.

V. SCREENSHOTS



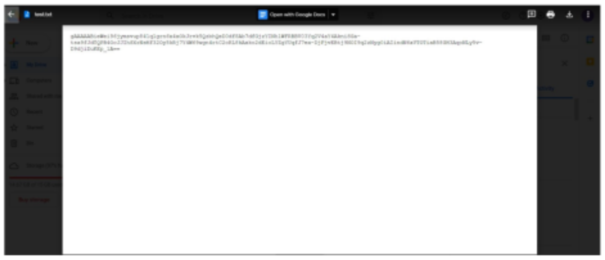
```
varshilrajeshbhainarola@Varshils-Air CS6301_src % python3 main.py
Enter login:varshil
Enter password:
Login successful!

upload - Uploading file , download - Download file, logout - Log Out , exit - Exit
upload

Ensure that files you wish to upload are in the 'F' folder.Enter return to return to main menu

Enter file name: test.txt
File ID:1G8PnGgmsnDTyaSfyDejXo4GsyZj9gGd-

Ensure that files you wish to upload are in the 'F' folder.Enter return to return to main menu

Enter file name: return

upload - Uploading file , download - Download file, logout - Log Out , exit - Exit
download

search - Searching file , download - downloading File ,exit - Exit
download
Enter file name: test.txt
Download 100%.

search - Searching file , download - downloading File ,exit - Exit
exit

upload - Uploading file , download - Download file, logout - Log Out , exit - Exit
logout
Log out
```

Fig. 4. User login interface



```
varshilrajeshbhainarola@Varshils-Air CS6301_src % python3 main.py
Enter login:admin
Enter password:
Login successful!

upload - Uploading file , download - Download file,usersettings -For User Setting, logout - Log Out , exit - Exit
usersettings

add - Add User , delete - Delete User , exit - Exit
add
Create login name:varshil_utd
Create password:
User created

add - Add User , delete - Delete User , exit - Exit
exit

upload - Uploading file , download - Download file,usersettings -For User Setting, logout - Log Out , exit - Exit
logout
Log out
```

Fig. 5. Admin interface for user add and delete



[illegible]

Fig. 6. Encrypted content of test.txt file

VI. CHALLENGES ENCOUNTERED

- User Authentication Complexity: Integrating user authentication with cloud services proved challenging. Nevertheless, the process was streamlined through the utilization of the Google Drive API and its comprehensive QuickStart Guide, which facilitated the implementation.
- Selection of Programming Language: Determining the appropriate programming language for cryptographic

operations was initially complex. However, extensive research led to the conclusion that Python's well-established libraries offer robust functionalities for cryptography.

- Encryption Algorithms: Identifying suitable encryption algorithms for both encryption and authentication presented difficulties. The project required a careful selection process to ensure that the chosen algorithms met security and performance requirements.
- Data Structure Design: Designing an effective data structure for securely storing usernames and passwords was a key challenge. It was essential to ensure that the structure was both secure and efficient for handling sensitive information.

VII. Improvements

The current implementation benefits from Google Authentication, which streamlines user access to the application. Future developments could include the creation of a graphical user interface (GUI) to enhance user interaction and experience by providing a more intuitive and structured approach. Additionally, improvements in thread and lock management could lead to better performance and reliability. Implementing a master server class could further optimize management of cloud storage contents and visitor statistics. For asymmetric encryption, incorporating external signature and verification processes could prevent unauthorized access and protect against public key theft.

VIII. Conclusion

In summary, the implementation of encryption for securing cloud storage is crucial for safeguarding sensitive information from unauthorized access and potential threats. This project successfully illustrates the deployment of a secure cloud storage solution leveraging public key cryptography, with asymmetric encryption used for secure key exchanges and symmetric encryption for data storage. The integration of the Google Drive API and Python libraries has significantly simplified the development of this solution. Looking ahead, future enhancements could focus on improving the user interface, system performance, and security features to further bolster the reliability and robustness of cloud storage solutions.

IX. References

References

[1] Junaid Hassan, Danish Shehzad, Usman Habib, Muhammad Umar Aftab, Muhammad Ahmad, Ramil Kuleev, and Manuel Mazzara, "The Rise of Cloud Computing: Data Protection, Privacy, and Open Research Challenges—A Systematic Literature Review (SLR)," Comput. Intell. Neurosci., vol. 2023, Article ID 9838129, 2023. This article has been retracted.

[2] Shweta Sukhram Prasad and Aarti Kanahiyalal Yadava, "Research on Cloud Data Storage Security," Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET), vol. 10, no. 6, pp. 1-9, June 2022. DOI: https://doi.org/10.22214/ijraset.2022.44694

[3] John W. Rittinghouse and James F. Ransome, *Cloud Computing: Implementation, Management, and Security*, CRC Press, 2009.

[4] Deepan Chakaravarthi Purushothaman and Dr. Sunitha Abburu, "An Approach for Data Storage Security in Cloud Computing," Int. J. Comput. Sci. Issues (IJCSI), vol. 9, no. 2, pp. 9-14, March 2012. License: CC BY-NC-ND 4.0.

[5] K. Roslin Dayana and P. Shobha Rani, "Secure Cloud Data Storage Solution with Better Data Accessibility and Time Efficiency," *International Journal of Cloud Computing and Services Science*, vol. 11, no. 2, pp. 756-763, May 2023. DOI: https://doi.org/10.1080/00051144.2023.2213564.

[6] J. Cai, Y. Hu, and Y. Li, "Research on the Method of Building a Secure Cloud Storage Platform," in *Proceedings of the 2022 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI)*, Zhuhai, China, 2022, pp. 17-20. DOI: 10.1109/IWECAI55315.2022.00011.

CS6301_Report.pdf

ORIGINALITY REPORT

15%	9%	7%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Trinity College Dublin Student Paper	3%
2	Submitted to Curtin University of Technology Student Paper	2%
3	"Cryptology and Network Security with Machine Learning", Springer Science and Business Media LLC, 2024 Publication	1%
4	Submitted to Liverpool John Moores University Student Paper	1%
5	Submitted to Australian National University Student Paper	1%
6	Submitted to University of Northumbria at Newcastle Student Paper	1%
7	i-scholar.in Internet Source	1%
8	gtusitecirculars.s3.amazonaws.com Internet Source	1%

9	www.tce.edu Internet Source	1 %
10	Submitted to Manipal University Student Paper	1 %
11	www.mjsat.com.my Internet Source	1 %
12	www.analyticsinsight.net Internet Source	1 %
13	iabac.org Internet Source	<1 %
14	Todor Stojanovski. "URBAN MORPHOLOGY AND CITY INFORMATION MODELLING (CIM) – COMPUTATIONAL URBAN DESIGN IN THE PLANNERS' OFFICE OF THE FUTURE", Contemporary Theory and Practice in Construction, 2024 Publication	<1 %
15	Ravi Das, Greg Johnson. "Testing and Securing Web Applications", CRC Press, 2020 Publication	<1 %
16	link.springer.com Internet Source	<1 %
17	swizex-email-sender.herokuapp.com Internet Source	<1 %
18	www.exeley.com Internet Source	

		<1%
19	fci.stafpu.bu.edu.eg Internet Source	<1%

Exclude quotes	Off	Exclude matches	< 3 words
Exclude bibliography	Off		