CSCI2660

May 14th, 2024 **CSCI2660 Capstone Dropbox** Ze Hua Chen

About the Project

The CS1660/CS2660 Computer Systems capstone project "Dropbox" involves creating a secure file storage system with an emphasis on security and design planning. This project was conducted in teams of two and includes the development of an API for secure file operations such as creating, authenticating, uploading, downloading, sharing, and revoking access to files.

Project Specifications

- Design Document: A detailed design plan outlining system's architecture, focusing on user authentication, file storage, sharing mechanisms, and file revocation procedures.
- Implementation: Following the design phase, we implemented our client in Python, ensuring the system's functionality and security as per the design document.

Functionality Overview

1. User Creation: Users provide a username and password, creating an associated struct stored in the Data Server. This struct includes encrypted fields like owned files, received files, and private keys.
2. User Authentication: To log in, the user provides their username and password. The system retrieves the user struct from the Data Server and uses the password and stored salt to generate a base key.
3. File Storage: Files are stored in 16-byte blocks, each encrypted and HMAC-ed using keys derived from a file base key.
4. Append File Efficiency: The *AppendFile* operation downloads only the metadata and the final block, avoiding the need to re-download the entire file. The new data is appended to the last block, and the updated blocks are encrypted and stored, maintaining efficiency.
5. File Sharing: Files are shared by encrypting the file base key with the recipient's public encryption key and storing it in the Data Server.
6. Revoking Shared Files: When access is revoked, the file is re-encrypted with a new file base key. The new key is shared with remaining authorized users, while the revoked user's access is removed.

Additional Capstone Specifications

Efficient File Updates: This feature enhances the performance of the *UploadFile* operation by enabling more efficient file component replacement. The client only uploads the blocks that have changed, reducing unnecessary data transfer.