Secure file-sharing service over Dropbox

Maeda Hanafi
Tai Liu
Nasser Zalmout

Dropbox

- Dropbox provides for efficient and scalable cloud file-sharing service.
    - Suitable for collaborative and global file access
    - Free, yet provides for cloud based scalability through premium access
    - Syncing functionality across different devices
    - Restore and backup functionalities.
- Dropbox is secure against external adversaries or eavesdroppers
- But what about internal access-permissions/security?
- How about access lists?

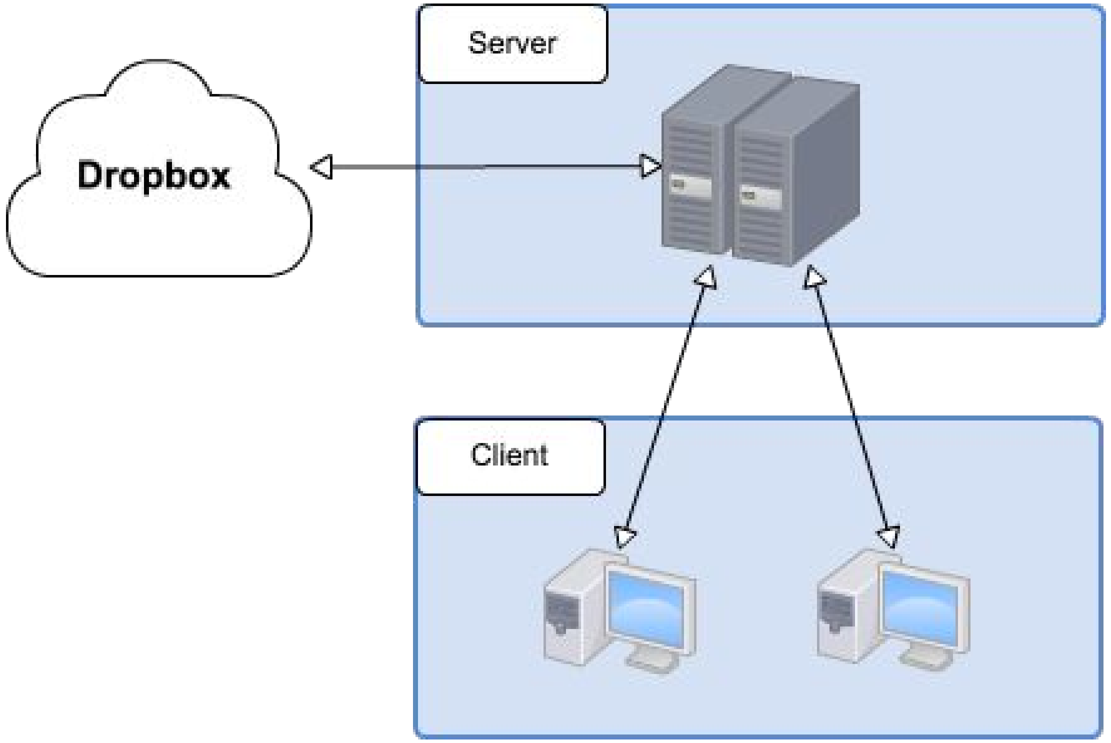
System layout

- Adding a secure layer on top of existing Dropbox functionality

- Server/client paradigm for authentication and access-list management

- Secure file sharing with several clients through an access list protocol.
  - Dropbox stores encrypted files only, using a symmetric key propagated at subscription
  - Files decrypted at download using the symmetric keys stored at the server’s access list.
  - Each file can be shared among several users through an access list managed by server for each file
  - All procedures are authenticated/managed through the server.
  - **Only** server has **direct** access to the dropbox folder

System design



Dropbox API

Several APIs for file management and sharing:

- Files upload/download:

  `files_upload`(*f*, *path*,mode), `files_download`(*path*, *rev=None*)

- Files search and deletion:

  `files_search`(*path*, *query*, max_results), `files_delete`(*path*)

- Files metadata:

  `files_get_metadata`(*path*,)

Overview of Security Defenses

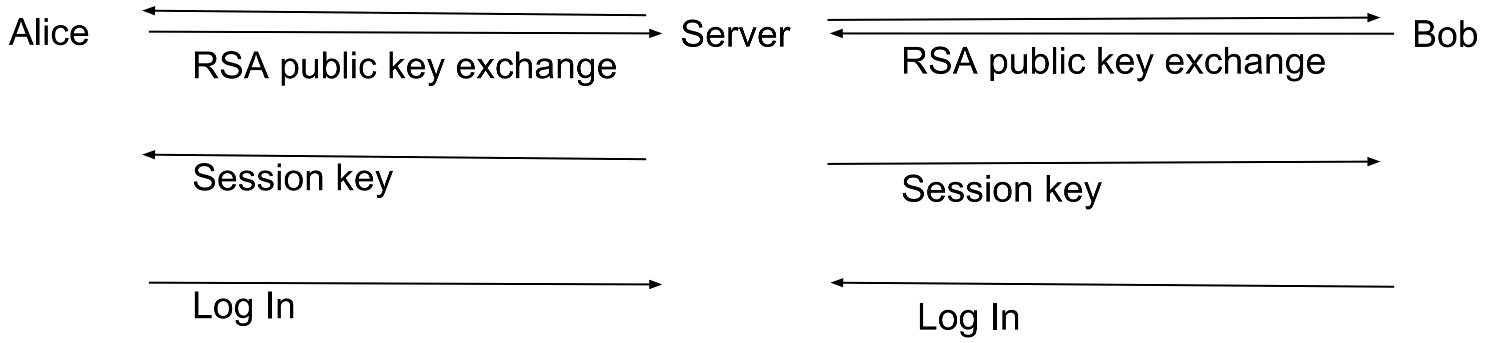
Defenses per Attack Types

- Eavesdropping
  - Encrypt all communication.
  - Establish unique session key for every new session between server and client.
- Tampering
  - Append hashed message authentication code (HMAC) after each message. The message receiver compares received HMAC against its own calculated HMAC.
- Replay Attack
  - Append timestamp to each message. The message receiver compares the received timestamp with the timestamp of the last received message.

Secure System Design - Logging In

Alice wants to send a file to Bob

Step 1: Alice and Bob establish their own session key with the server using RSA. Session key is used as the key for future communication.



Secure System Design - Uploading

Step 2: Alice sends the file to the server, and the server sends the file to Dropbox. The file is encrypted on Dropbox, including the filename. E() is the encryption function.

$$\begin{tikzcd}[column sep=huge] \text{Alice} \arrow[r, "\text{Sends E(File, Session key)}"'] & \text{Server} \arrow[r, "\begin{array}{l}\text{Metadata = (Filename, File, Allowed Users)} \\ \text{Sends E(Metadata, Unique file key)}\end{array}"'] & \text{Dropbox} \end{tikzcd}$$

Secure System Design - Downloading

Step 3: Bob requests to download the file from the server. The server retrieves it from Dropbox and sends it to Bob.

$$\begin{tikzcd} \text{Dropbox} \arrow[r, "\begin{array}{l}\text{Metadata = (Filename, File, Allowed Users)}\\ \text{Sends E(Metadata, Unique file key)}\end{array}"'] & \text{Server} \arrow[r, "\text{Sends E(File, Session key)}"'] & \text{Bob} \end{tikzcd}$$

Up to now, it looks like a secure email system....

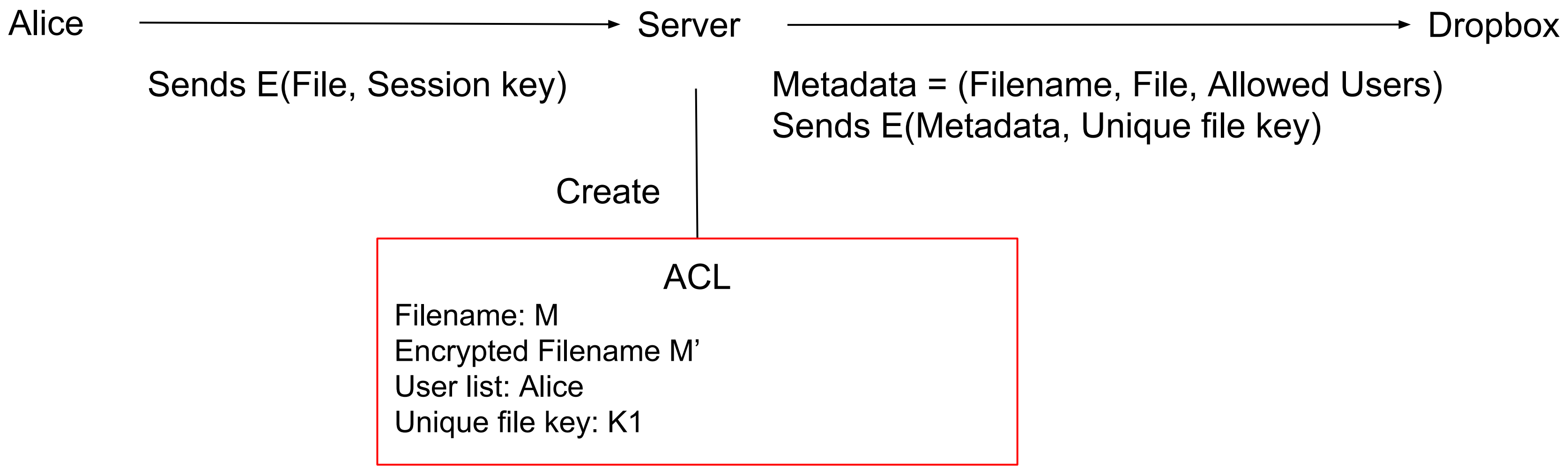
But we want to make a **secure file-sharing service** application. How can we achieve that?

Secure File Sharing - Access Control List (ACL)

- Filename
- Encrypted Filename
- User List
- Unique File Key

Secure File Sharing - Uploading

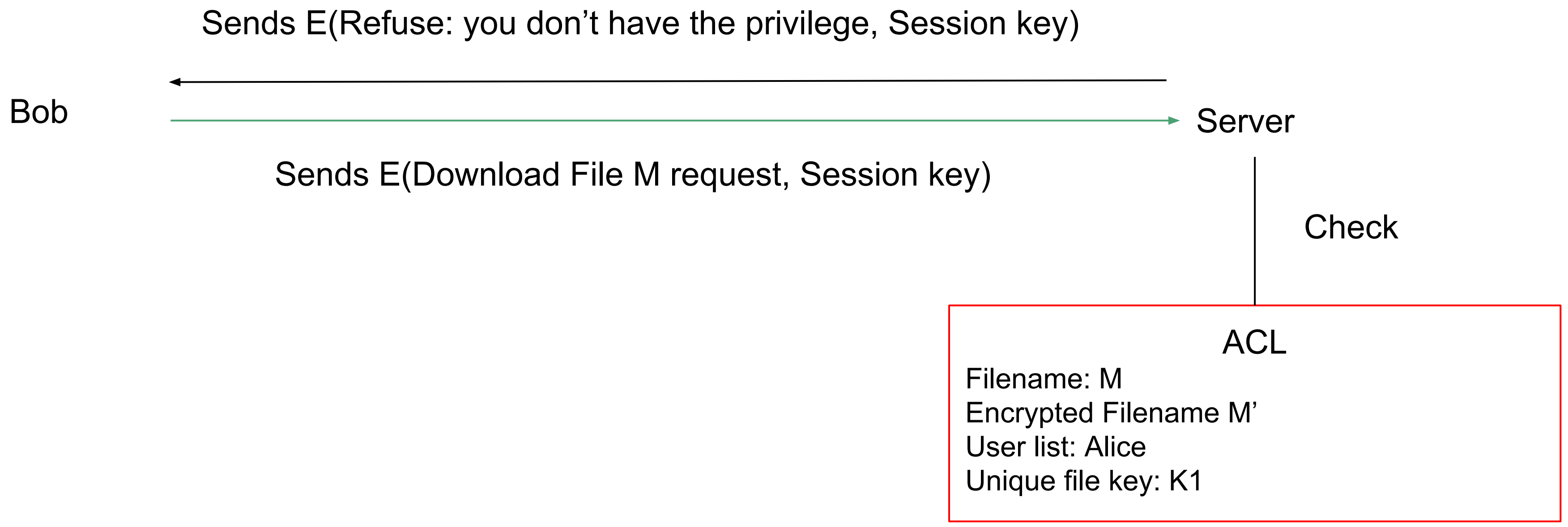
Alice sends a file M to the server, and the server sends the file to Dropbox. The file is encrypted on Dropbox, including the filename. E() is the encryption function.


Alice
Server
Dropbox
Sends E(File, Session key)
Metadata = (Filename, File, Allowed Users)
Sends E(Metadata, Unique file key)
Create
ACL

Filename: M
Encrypted Filename M’
User list: Alice
Unique file key: K1

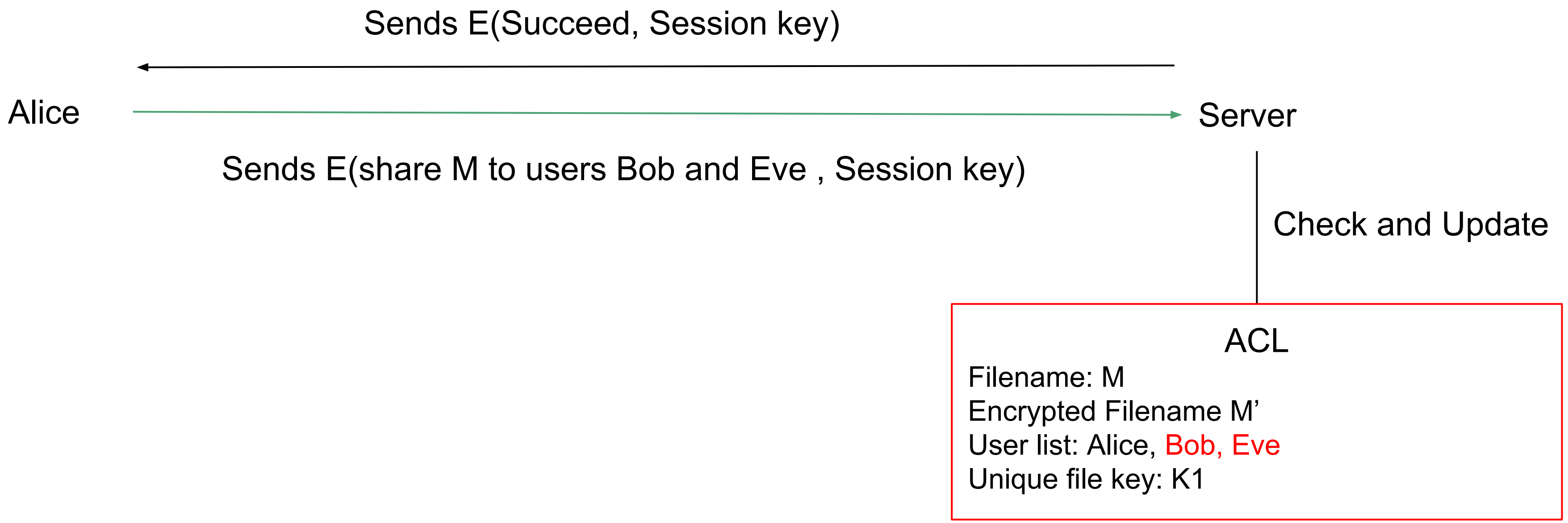
Secure File Sharing - Downloading

Bob requests to download the file M from the server. The server retrieves it from Dropbox and sends it to Bob.


Sends E(Refuse: you don’t have the privilege, Session key)
Bob
Server
Sends E(Download File M request, Session key)
Check
ACL
Filename: M
Encrypted Filename M’
User list: Alice
Unique file key: K1

Secure File Sharing - Add users

Alice wants to share the file M to Bob and Eve.


Sends E(Succeed, Session key)
Alice
Server
Sends E(share M to users Bob and Eve , Session key)
Check and Update
ACL
Filename: M
Encrypted Filename M'
User list: Alice, Bob, Eve
Unique file key: K1

Secure File Sharing - Commands Implemented

- Upload a file and share it to some users
- Download a file
- Add users
- Delete users
- List files
- Delete a file

All requests must be first checked against ACL by the server

Demo

Check the video at the following link for a detailed operation flow:

https://drive.google.com/a/nyu.edu/file/d/0BzAXNUTAvnZ9UWZWaHFJR3ILU2c/view?ts=57262e9f

Thank you for your attention!