Telecommunication - Securing the Cloud Server

Kmla Sharma, 13319349

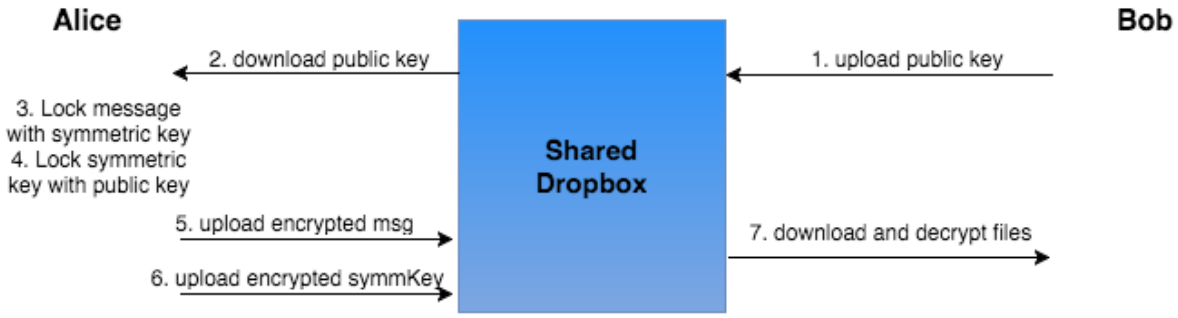
Introduction

This report will outline the description and implementation of creating a java application to manage and secure selected files before they are uploaded to a shared dropbox account. This involves encrypting the file and managing the various keys generated, and a combination of asymmetric and symmetric algorithms.

Design and Implementation

1. Key Management

For example, if Alice wants to send Bob (who is an authorised member of her dropbox group) an encrypted message, the following exchange will occur:



1. Sending the Source File

In order to maximise efficiency and reduce overhead management of keys, the source file (ie the message to send) is encrypted using a symmetric algorithm, such as AES. This involves the creation of only one key to lock and unlock the file with, and reduces encryption time if the file to send is particularly large. However, this symmetric key must be encrypted prior to sending the encrypted source file, to prevent eavesdroppers from obtaining they key and decrypting the file.

2. Sending the Symmetric Key

Once a symmetric key for the source file has been generated, it must be encrypted asymmetrically, eg. the RSA encryption algorithm. This particular algorithm generates two keys, a public key, responsible for encryption, and a private key responsible for decryption. In the above scenario, Bob's public key, which is shared and open to everyone, is used by Alice to encrypt her symmetric key. Bob retains his private key separately, away from the dropbox cloud

storage, which is then used to the decrypt the symmetric key. In this scenario, only Bob is able to decrypt the key, as his private key is unshared.

2. User management

Once the dropbox account has been successfully linked with the java application and an admin has been established, they are free to authenticate (add) a user to the dropbox storage to communicate with.

This involves creating a type of simple log in system, whereby many users can be added to the group, however before the admin can actually initiate communication with them (i.e. receive their public key to begin the encryption process), the user must log in with their designated pin code, distributed to them via email when they are initially added to the group. Once correctly authenticated, the communication process can begin. As the admin is primarily in charge of initiating communication, they are only able to do so with the authenticated users, and are unable to see/communicate unauthenticated users. This minimises risk of communicating with attackers/intruders. Authenticated users are kept track of using a Hashtable, containing their name and their allocated pin code.
Deleting a user is also a simple process, whereby their pin and name are deleted from the Hashtable and therefore preventing any communication.

Conclusion

I feel that my approach is very secure and efficient, as it accounts for all scenarios, such as an attacker attempting to to decrypt the key or unauthenticated users trying to communicate with the admin.