Secure File System

Kuldeep Kumar Solanki(19111045), Prashant Piprotar(19111063)

September 1, 2019

Contents

1 INTRODUCTION

Now a days security is a big concern in file storage over the Public Cloud. It's not easy to trust online clouds like Dropbox, google drive for storing our sensitive data. So, that's where we came with an idea of SECURE FILE STORAGE. In this we have our own custom file system, where we can store easily and share securely. In the base of the system we are only using standard cryptography algorithms like RSA, SHA256 to maintain the transparency of the design.

2 DESIGN

We have divided our system into basic functional modules. Our implementation of system will revolve around this properties :

2.1 Initialization of user

We are creating a user data structure in which we are storing mainly username, password, private key, file map type and sharedfile map. File map contain file pointers. By using username we are going to generate RSA key and then store it in the keystore. We are generating the argon2key using combination of username and password and encrypt our user data structure using "CFB Encryption" system. We will also generate the mac of user data structure to check integrity. The encrypted data we will store in the Storage Server.

2.2 Getting user information

We are decyrpting the user data structure using Argon2key which we we will again generated with combination of username and password. We will calculate the hash of received data structure and compare it with the already given hash, if it matched then integrity accomplished. After that we will decrypt our user data structure and populate in our structure.

2.3 Storing user data

We are going to use map like file structure to store the files on the storage server with the root node/metadata placed at user data structure in map data type. Each file block will be first hashed using SHA256 and then encrypted by separate key(generated by argon2) using our CFB encryption. Further the blocks will be pointed by the metadata of the file, which will reduce the handling overhead from user data structure. At last the block size will be provided by a global variable. The file size given for storing will be in the multiple of the block size, if not provided it should generate error.

2.4 Append file

In append file function we have to open up the user data structure to look at the map of the file and its pointer to the metadata. After getting the metadata, it will be iterated to the last block and start appending the new blocks subsequently (including hashing and encryption of those blocks). At last, append data has to be in the multiple of the block size.

2.5 Retrieve file

Retrieve file will be done by load file function. During retrieval of the file as block by block, it will contain the hash (each block) to check the integrity and will be decrypted block by block. The file will be loaded as per the offset given by the function call.

2.6 Sharing of file

The share function of the file is simple in compare to others. It will pack the part of the metadata like key and location of the file in the message and encrypt it with the public key of the receiver using RSA alogorithm. The receiver will decrypt the message using its private key. Then it will store the file with the name whatever it wants. The file can be shared between multiple users.

2.7 Revocation of access

Revocation of access will be done only by the root user/primary user of the file. It will revoke the access of the file from other user by loading all the data and re-encrypting with the new key, so that other users can't access it further.

3 CONCLUSION

Our system ensures standard security of the files based on minimal functionalities.