Table 1 User Properties

Category	Property	Symbol	Purpose
Keys	master key	mk	symmetric key, serves as the root key of the user's file system which will be further illustrated in Section 4.
	public key	pk	asymmetrick key pair, used to send and receive secret information over the untrusted server or open channel which will be further illustrated in Section 3.5
	private key	rk	

we will $E_{key}(data)$ and $D_{key}(data)$ to denote encryption and decryption data with respective key throughout the paper.

4 USER MANAGEMENT

4.1 User Property Set

User property set is the collection of properties associated with a particular user that will be used to fulfill the functions provided by SafeStorage as indicated in Section 3. Obviously for any system with multiple users and authentication mechanism, the username and password are the most important property for a user, which is represented by u and p respectively. For SafeStorage, other properties are designed to achieve the functionalities provided and they are summarized in Table 1.

In this paper, we use {} to represent the property set of a user so for user U we have $U = \{u, p, mk, pk, rk\}$. We also use $U.property\_name$ to represent user U's certain property like $U.pk$ represents U's public key.

4.2 User Registration

User registration is the first step before a user can enter the system and also the time when all the initialization works, both on client side and server side, are settled. Figure 1 illustrates the protocol between client side and server side for user setup.

In order to facilitate the understanding of the user registration protocol, we would like to highlight some design logic here:

(1) All user properties are created at client side without server involvement; i.e., the server has no control on the key generation step and has no way to guess or derive the keys generated.

(2) The ideal situation is to store all the keys on the server. However, as we assume the server is untrusted, if a user stores the whole set of keys on the server, then the server is able to combine all the information to derive certain keys and then decrypt data the user stored, which means users' data privacy is compromized. Therefore, we require that rk must be kept at the client side while other user properties can be stored on server after some transformation.

4.3 User Authentication

Authentication is the process when user proves his identity to the server. In our system, we assume the normal procedure for user authentication.

A user sends his username and hashed password to the server and the server authenticate the user by matching the received username and password hash with the existing record stored in database. If there is a match, the user's identity is proved and the server returns login success information, otherwise, returns login failure information.

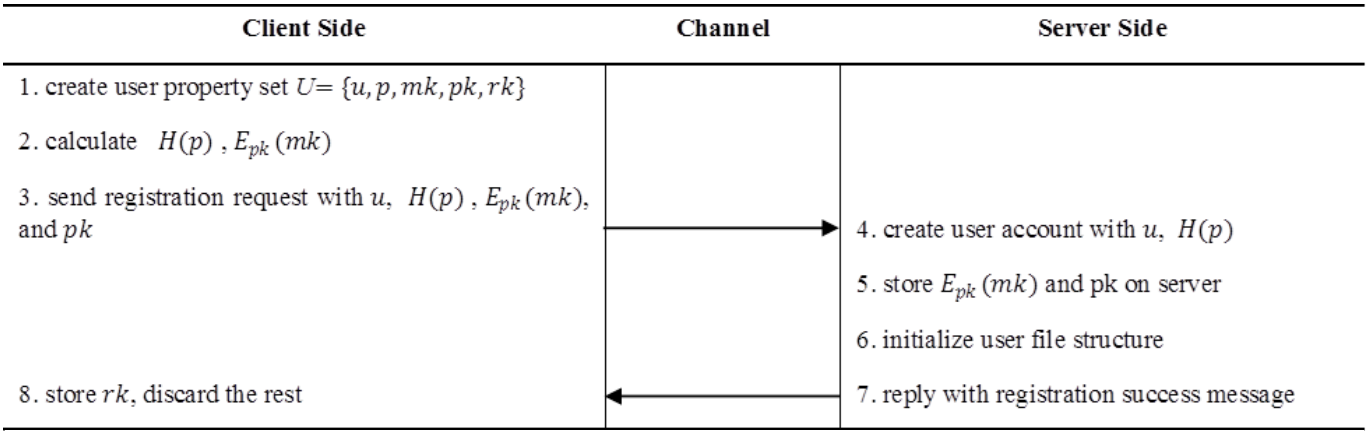


Figure 1 User Registration Protocol

4.4 Key Protection under Open Channel

Under the condition of open and insecure communication channel, people may worry that some malicious party may spy over it and get the username and hashed password information from the channel. Any adversary who obtains these information is able to perform replay attack by maliciously or fraudulently repeating sending these information to the server to pretend the identity of the user. However, in the design of SafeStorage, as the rk is always kept at the user's client side devices, even if any adversary successfully performs the replay attack and obtains the key information stored on the server; i.e., $E_{pk}(mk)$ and pk, without rk, the adversary is still unable to decrypt $E_{pk}(mk)$. Therefore, mk, the master key for the file system, is still protected. Obtaining pk does not help to decrypt any information nor deriving rk and thus, is useless to the adversary. This is also the reason why pk can be stored on the server in plain text.

However, if the adversary somehow obtains the rk from a user's client device, then the adversary is able to derive mk from the key information stored in server. Protecting data in a user's client device is out of the scope of the paper and there are various softwares and mechanisms to help achieve this.

5 A FILE SYSTEM FOR SECURE STORAGE AND SHARING

5.1 Notations

1 Key Link

Notation: Given two secret keys K_1 and K_2 and a symmetric encryption scheme, we denote the result of encryting K_2 with K_1 by $K_1 \rightarrow K_2$, we call this a key link, meaning we can derive K_2 if we knows K_1 and the encryption scheme.

2 File and File Key

In this paper, we use $\mathcal{F}$ to represent a file and we use $\mathcal{F}.fk$ or $fk_{\mathcal{F}}$ to represent $\mathcal{F}$'s file key and $\mathcal{F}.par$ to represent $\mathcal{F}$'s parent folder (if $\mathcal{F}$ is not the root folder). The root folder is denoted by R.

5.2 Basic Structure

In order to build an intuitive and efficient access control scheme for the purpose of file sharing, we present a key management scheme in the form of a key tree with a similar structure of the file tree. Following are the definitions of the two trees:

(1) **File tree**: a tree that reflects the logical organization of the files in a file-folder structure, which is also the basic structure for the other two trees. Changes in file tree will be reflected in the other two trees.

(2) **Key tree**: a tree that manages all the keys used in the file system to provide secure and efficient file storage and sharing.

Figure 2 shows a sample file system for one user as well as the relationship between the two trees and detail discussions are provided in later part of this chapter.

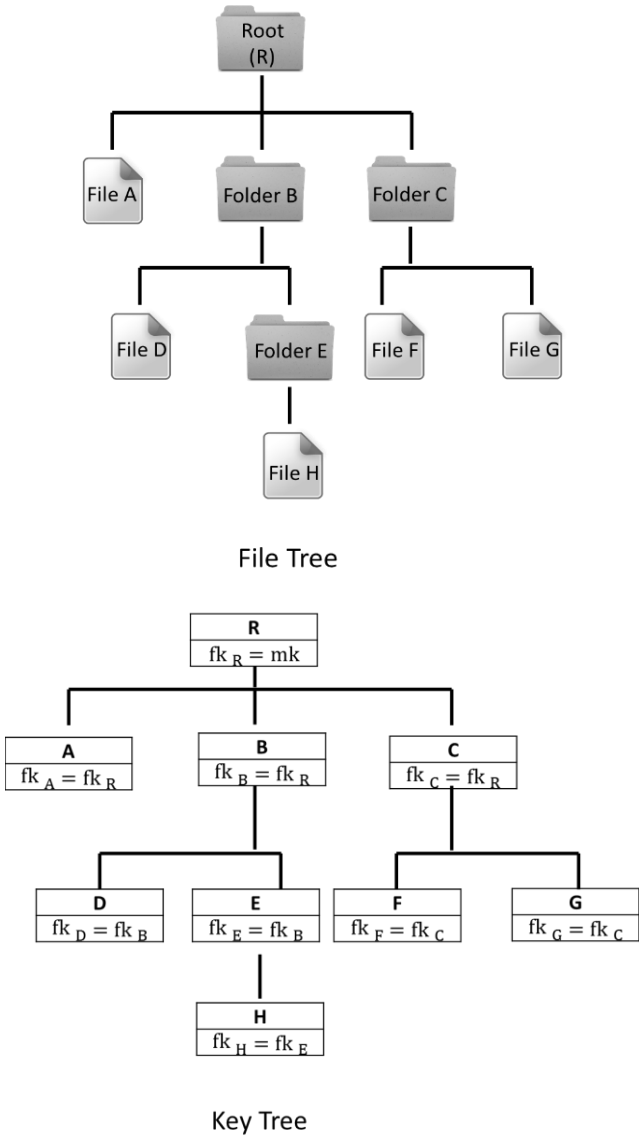


Figure 2 File Tree and Key Tree in Initial State

Intuitively, in terms of sharing and unsharing files/folders to other users, the access rights are granted and revoked based on the level of the file

in the tree. In other words, if a folder, say Folder B is shared to another user, say Alice, Folder B as well as all the subfiles and subfolders (i.e., File D and Folder E) are shared to Alice too. Similarly, if there are new files or folders added under Folder B, these new files or folders are available to Alice too by default. And if files or folders are moved out of Folder B to other places on the system (not deleted), access rights to these files will automatically be revoked.

Figures 2 also presents the initial state of a user's file system, the state when only the user is accessing his files and folders and no file sharing instructions have been issued.

5.3 Key Management for File Sharing

5.3.1 Procedure for Granting Access Rights (File Sharing)

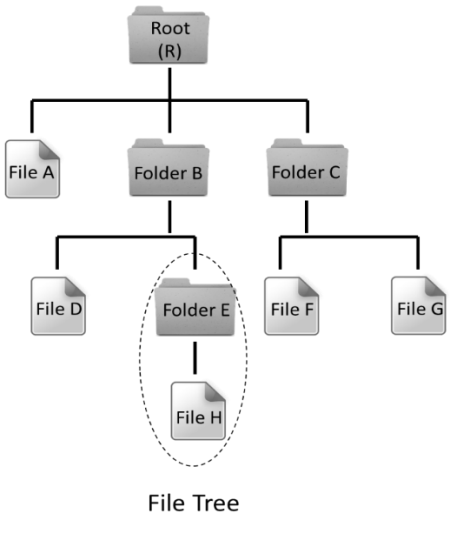
In SafeStorage, the procedure to grant access rights of a particular file/folder to other users is stated in the scenario analysis. We will discuss the file sharing procedure in 3 scenarios and see how file tree and key tree are modified to handle access control for file sharing:

(1) Share a file/folder with user X
(2) Share a folder with user X which is inside a shared folder with user Y
(3) Share a folder with user X which includes a shared file/folder with user Y

Note in the diagrams, keys and hash values in shaded areas are the affected items due to this file sharing operation.

1. Share a folder with user X

Figure 3 shows the changes in the 2 trees when the user desides to share folder E with X.



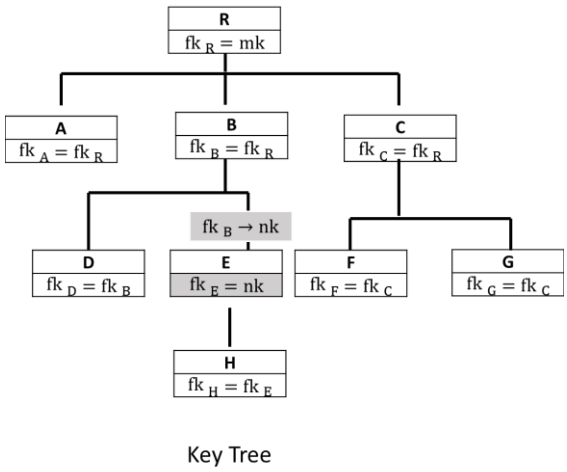


Figure 3 Share a folder with user X

2. Share a folder with user X which is inside a shared folder with user Y

Figure 4 shows the changes in the 2 trees when the user desides to share folder E with X whose parent folder B has been shared with Y previously.

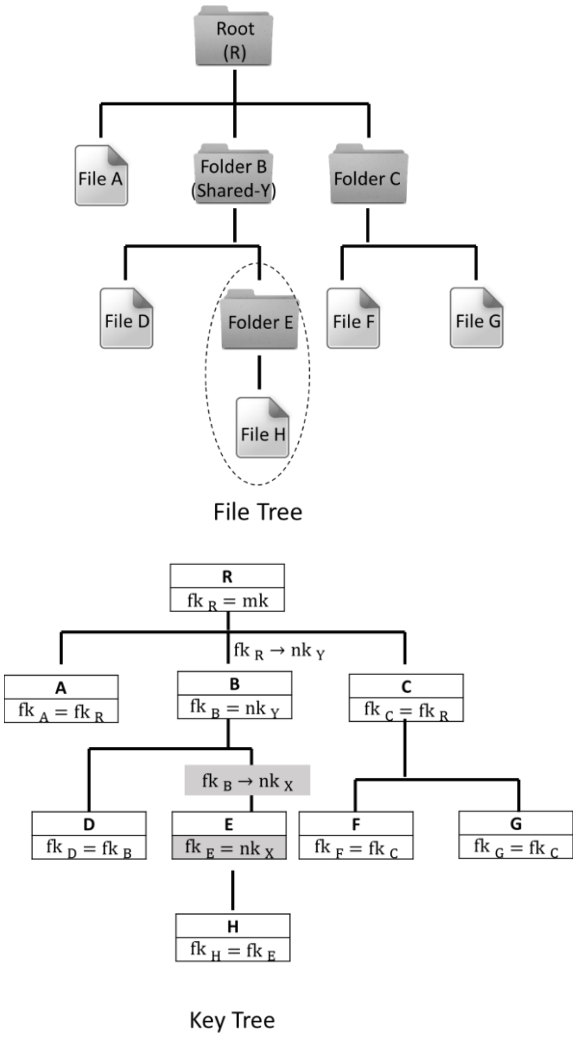


Figure 4 Share a folder with user X which is inside a shared folder with user Y

3. Share a folder with user X which includes a shared file/folder with user Y

Figure 5 shows the changes in the 2 trees when the user desides to share folder B with X whose subfolder E has been shared with Y previously.

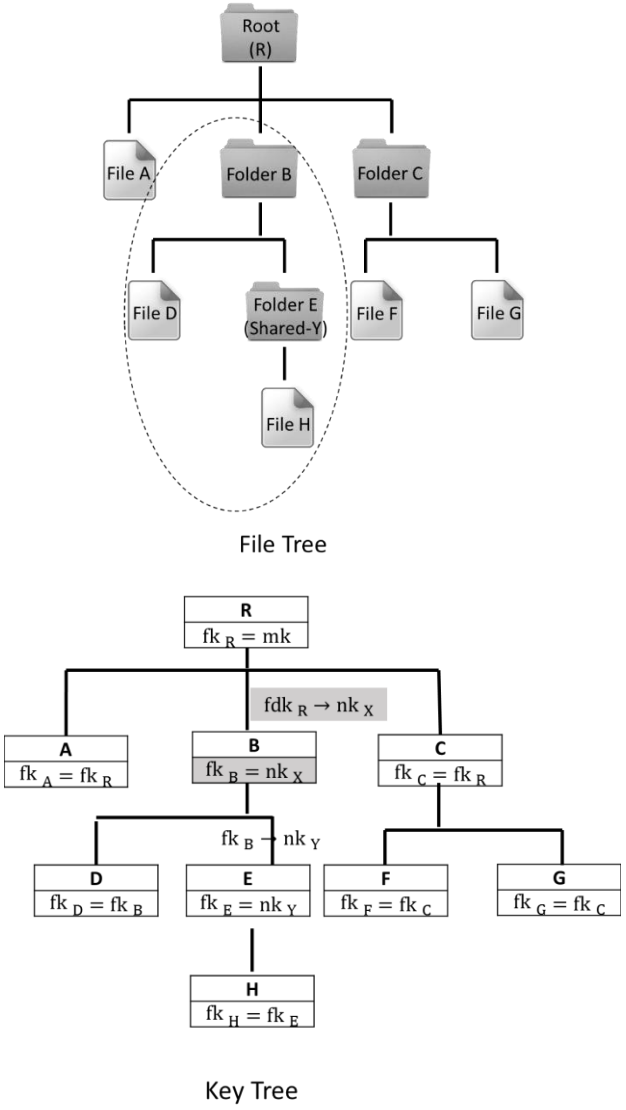


Figure 5 Share a folder with user X which includes a shared file/folder with user Y

In summary of the 3 scenarios, if $\mathcal{F}$ has been shared with some users, then send $\mathcal{F}.fk$ to each user that is granted access rights securely; otherwise, create a new key nk, re-encrypt $\mathcal{F}.fk$ with it coupled with necessary updating of hash tree. And then set $\mathcal{F}.fk$ to be nk and share $\mathcal{F}.fk$ to each user that is granted access rights securely.

5.3.2 Procedure for Revoking Access Rights (File Unsharing)

In SafeStorage, the procedure to revoke the access rights of a particular file/folder from other users is stated in the following algorithm. We will discuss the file unsharing procedure in 2 scenarios and see how file tree and key tree are modified to handle access control for file sharing:

(1) Unshare a folder with user X which is inside a shared folder with user Y
(2) Unshare a folder with user X which includes a shared folder with user Y

Note in the diagrams, keys and hash values in shaded areas are the affected items due to this file sharing operation.

1. Unshare a folder with user X which is inside a shared folder with user Y

Figure 6 shows the changes in the 2 trees when the user desides to unshare folder E with X whose parent folder B has been shared with Y previously.

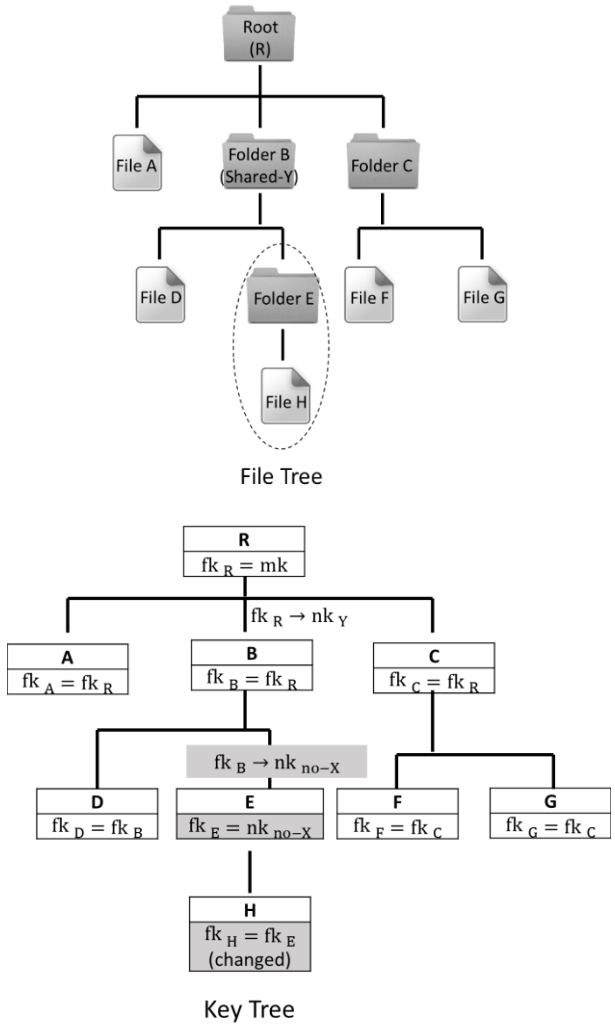


Figure 6 Unshare a folder with user X which is inside a shared folder with user Y

2. Unshare a folder with user X which includes a shared folder with user Y

Figure 7 shows the changes in the 2 trees when the user desides to unshare folder B with X whose subfolder E has been shared with Y previously.

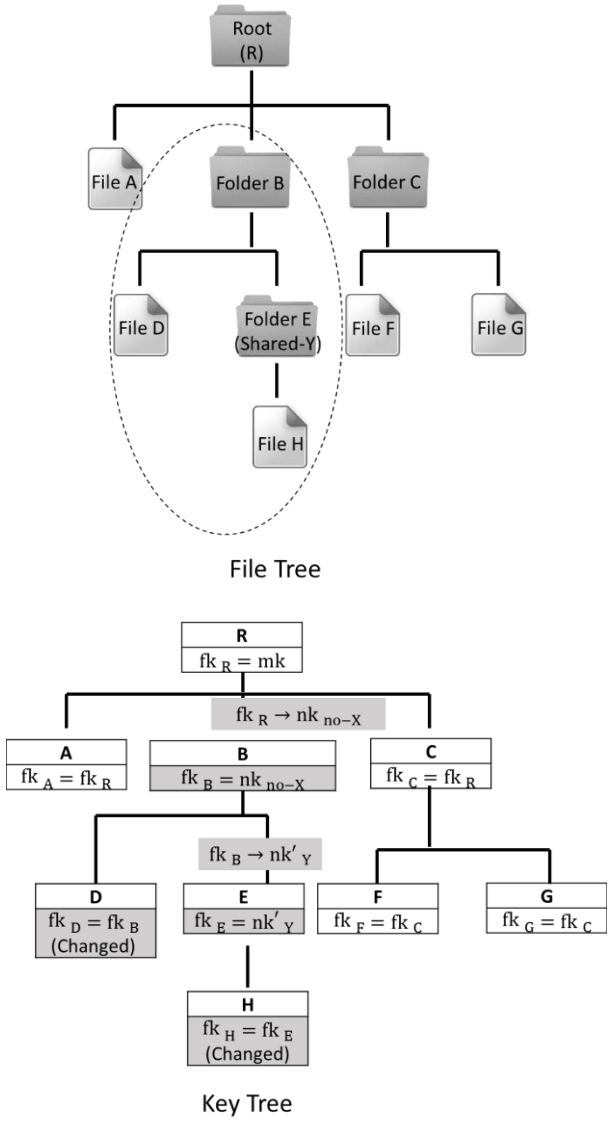


Figure 7 Unshare a folder with user X which includes a shared folder with user Y

In summary of the 2 scenarios, if the access rights for file/folder $\mathcal{F}$ is revoked for a group of users, the key updating procedure is a cascading procedure from top to bottom. If $\mathcal{F}$ is a file, then generate a new key nk, update the file key $\mathcal{F}.fk = nk$ and re-encrypt $\mathcal{F}$ data with nk. Otherwise if $\mathcal{F}$ is a folder, then generate a new key nk and set $\mathcal{F}.fk = nk$. For any the subfile or subfolder $\mathcal{F}'$ of $\mathcal{F}$, generate new file key nk' for $\mathcal{F}'$ and re-encrypt $\mathcal{F}'$ with nk'. If if there exist key link between $\mathcal{F}$ and $\mathcal{F}'$, re-encrypt the key link with nk; i.e. $\mathcal{F}.\text{fk} \rightarrow \mathcal{F}'.\text{fk}$ becomes $nk \rightarrow nk'$. If $\mathcal{F}'$ is a file, the process can stop, otherwise the revocation process goes recuisively with $\mathcal{F}'$ until reaching bottom.

5.4 File System Operations

In this section we discuss how basic file system operations can be carried out in SafeStorage and how the operations can impact the key tree in the system. The scenarios discussed in this section include addition, deletion, move and copy operations on files or folders.

1. File/Folder Addition/ Deletion

To add a file/folder $\mathcal{F}$ to the given file system only one action must be taken which is to link the file in key tree by setting $\mathcal{F}.fk = \mathcal{F}.par.fk$.

Similarly, for file/folder deletion, the only action is to unlink the file in the key tree by deleting file $\mathcal{F}$ and any key link between $\mathcal{F}.par$ and $\mathcal{F}$.

Figure 8 shows the file addition operation:

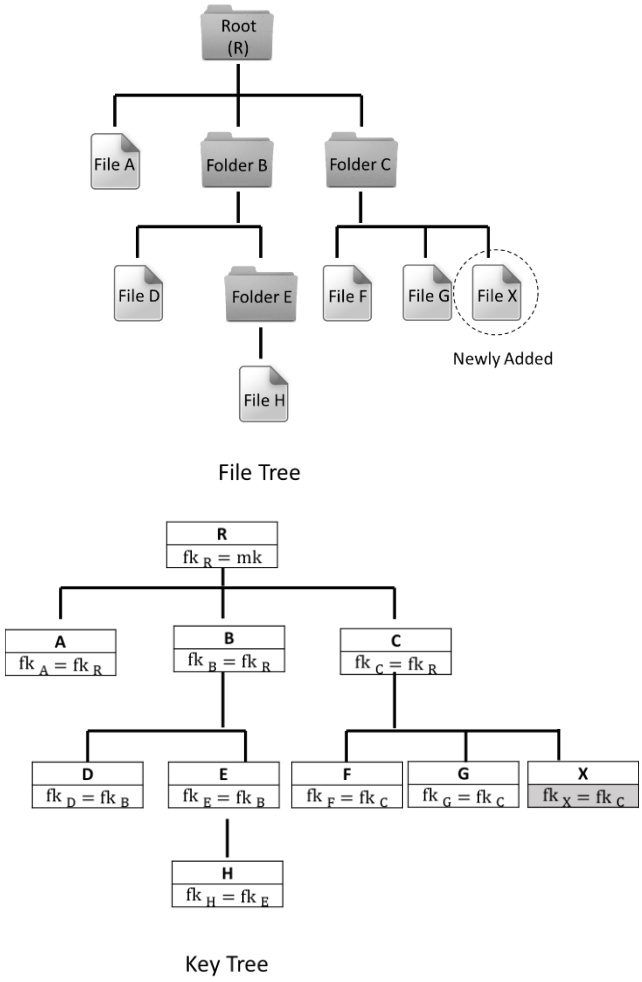


Figure 8 File Addition Operation

2. File/Folder Copy/Move

For file/folder copy, there is no action required for the original copy $\mathcal{F}$ but for the new copy $\mathcal{F}'$ several actions must be taken: (1) retrieve $\mathcal{F}'$ parent folder key $\mathcal{F}'.par.fk$ and re-encrypt $\mathcal{F}'$

with it (2) link the file in key tree by setting $\mathfrak{F}'.fk = \mathfrak{F}'.par.fk$.

For file/folder move, the file delete operation must be applied on the original copy $\mathfrak{F}$ and for the new copy $\mathfrak{F}'$ similar actions like the $\mathfrak{F}'$ in file copy must be applied. This makes file move the most complicated operation on the file system.

Figure 9 shows the file copy operation:

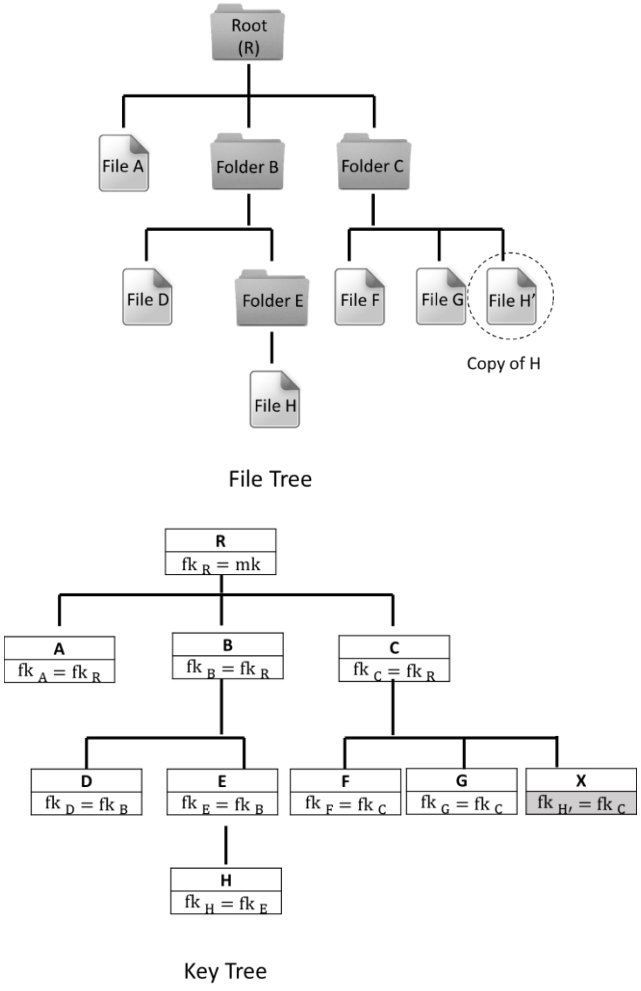


Figure 9 File Copy Operation

6 CONCLUSION

We believe that SafeStorage provides a simple and basic solution to build a file storage and sharing system on an untrusted cloud and in an open and insecure communication channel. We have proposed several handy but efficient cryptographic solutions to address various problems arising in the operations of such a system.

The strengths of SafeStorage are its simplicity, efficiency, flexibility and readiness to implementation. Thus, SafeStorage can serve as a good foundation for development of advance features like group collaboration and versioning and recovery. The file management and sharing system which is co-supported by 2 trees is also expected to be used in the future development phrase.

In future work, additional techniques are investigated to further improve the simplicity and efficiency of SafeStorage, in particular the separation of read/write access rights for file control as discussed in [17][18][21]. Furthermore, the performance of SafeStorage needs to be properly tested and compared with some existing online storage service providers which do not require client side encryption/decryption or use different approaches.

REFERENCES

[1] J. Blackwood, "Is storage outsourcing a viable alternative?," [Online]. Available: http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2851289,00.html.

[2] JustCloud, "How JustCloud Works," [Online]. Available: http://www.justcloud.com/how-it-works.

[3] Dropbox, "Always Be Safe," [Online]. Available: https://www.dropbox.com/tour/4.

[4] BoxCryptor, "Introduction," [Online]. Available: https://www.boxcryptor.com/.

[5] CloudFogger, "CloudFogger - Introduction," [Online]. Available: http://www.cloudfogger.com/en/.

[6] Wuala, "Security & Privacy," [Online]. Available: http://www.wuala.com/en/learn/features/t/2.

[7] Mega, "Mega - Introduction," [Online]. Available: https://mega.co.nz/.

[8] K. Fu, "Group Sharing and Random Access in Cryptographic Storage File Systems," *Master's Thesis, Massachusetts,* 1999.

[9] K. Fu, S. Kamaram and Y. Kohno, "Key Regression: Enabling Efficient Key Distribution for Secure Distributed Storage," *Computer Science Department Faculty Publication Series,* 2006, 149.

[10] M. Backes, C. Cachin and a. A. Oprea, "Secure Key-Updating for Lazy Revocation," *11th European Symposium On Research In Computer Security (ESORICS),* pp,327-346, 2006.

[11] H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," *Internet Engineering Task Force (IETF)*, 2010.

[12] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0," *Network Working Group*, 2000.

[13] D. Bindel, M. Chew and a. C. Wells, "Extended cryptographic file system," *Unpublished manuscript,* 1999.

[14] E. Zadok, I. Badulescu and a. A. S. Cryptfs, "A stackable vnode level encryption file system," V184, Technical Report CUCS-021-98, Columbia University, 1998.

[15] E.-J. Goh, H. Shacham, N. Modadugu and D. Boneh, "SiRiUS: Securing remote untrusted storage," in *Proceedings of NDSS.*, 2003.

[16] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang and K. Fu., "Plutus: scalable secure file sharing on untrusted storage," in *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST)*, pp. 29-42, 2003.

[17] G. Cattaneo, L. Catuogno, A. D. Sorbo and a. P. Persiano, "The design and implementation of a transparent cryptographic file system for UNIX," in *Proceedings of USENIX Technical Conference, FREENIX Track*, pp. 199-212, 2001.

[18] D. Mazi`eres, M. Kaminsky, M. F. Kaashoek and E. Witchel, "Separating key management from file system security," in *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles (SOSP)*, pp. 124-139, 1999..

[19] D. P. O'Shanahan, "CryptosFS: Fast cryptographic secure NFS. Master's thesis," University of Dublin, 2000.

[20] E. Geron and A. Wool, "CRUST: Cryptographic Remote Untrusted Storage without Public Keys," in *Proceedings ofSecurity in Storage Workshop, 2007. SISW '07. Fourth International IEEE*, pp. 357-337 2007.

[21] M. V. Dijk, L. F. G. Sarmenta, J. Rhodes and S. Devadas, "Securing Shared Untrusted Storage by using TPM 1.2 Without Requiring a Trusted OS," *MIT CSG Memo 498,* 2007.

[22] D. Grolimund, L. Meisser, S. Schmid and R. Wattenhofer, "Cryptree: A folder tree structure for cryptographic file systems," *Reliable Distributed Systems,* pp. 189-198, 2006.

[23] A. Zych, P. Milan and W. Jonker, "Efficient key management for cryptographically enforced access control," *Computer Standards & Interfaces*, pp. 410-417, 2008.

[24] J. Lin, K. Huang, F. Lai and H.C. Lee, "Secure and Efficient Group Key Management with Shared Key Derivation," *Computer Standards & Interfaces*, pp. 192-208, 2007.