Cryptographic File System: Easy and Reliable?

Mohammad Reza Abbasy, Mahdi Sharifi, and Mohammad Reza Najaf Torkaman

***Abstract*—Complications to encrypt files and check the vulnerabilities every time had been a crucial and lasting so far. So what is the trustable and feasible approach for the purpose of manifesting objectives? Developing file system with cryptographic features can promote liabilities. The Cryptographic File System (CFS) provides a secure and reliable storage by using UNIX file system for encrypting files. Utilizing a key for encrypting and decrypting directories and files (also their pathnames) is usual in any cryptography issue and in this paper we also mention about it. System administration process like backup, establish in a suitable way without knowing about the key. Ability to respond any accessible file system without any modification in order to cover remote servers like NFS is another advantage of CFS. This paper explains the CFS under UNIX briefly which is also applicable in UNIX like system such as Linux.**

***Index Terms*—Cryptography, cryptographic file system, file encryption.**

I. Introduction

Nowadays, security is a crucial issue especially in modern computing systems. Sometimes, scientists remember it as a difficulty. Intruder has a very prominent role in security's area. They are able to disturb any stable and convenient condition in order to gain access to classified information in both of small or large systems. Most of the time, security administrator lay stress on saving the sensitive data in a computer which is not connected to any network because if intruder can obtain an unauthorized access he or she can do anything in harmful way. So availability of systems must be emphasized by adopting some techniques and policies. For example make backup from our information. The main goal of cryptographic methods intends to protect data in front of unauthorized users. Some necessities must be established to achieve protection entirely.

Utilized, new cryptographic algorithms like RSA [1], DES [2], and IDEA [3] are fully trusted to comply with security goals. Unfortunately, using modern cryptographic algorithms is not customary to preserve files because in some cases algorithms are not available for anyone or they are very difficult to use and user must aware scientific process. So, what is the best way to gain high security without involving extra difficulties? How we will able to manage popular misconceptions or human faults totally and in details? This paper wants to express, cryptographic file system is better perspective to solve common and expert problems for protecting information.

Manuscript received March 14, 2013; revised September 8, 2013.

The authors are with Advanced Informatics School (AIS), International Campus, Universiti Teknologi Malaysia (UTM), Kuala Lumpur (e-mail: amohammad2@live.utm.my, smahdi2@live.utm.my, rntmohammad2@live.utm.my).

II. Levels of File System Cryptography

From the point of view of file system cryptography, it falls into two categories; user and system level cryptography. In action, each of them has advantages and disadvantages. In user level file system cryptography, user utilizes software like UNIX and Linux [3] crypt tool to encrypt or decrypt file with a key. So in this type of user level file system cryptography, user must handle the coding and decoding the file. It was first approach. In second manner, there is a built-in cryptography facility in application software. For instance, a text editor can read or write from disk by asking from user to import a key for reading or writing. On the other hand, when the editor writes to the disk, it is encrypting and when it is reading from disk, it is decrypting the file. Unfortunately, both of them are not useful in order to meet complete security, totally. For example, the former method contains human fault while it uses a very convenient way to encrypt and decrypt the file; the user might unintentionally forget to delete original text after coding. Also, every time the user wants to encrypt or decrypt the file, he or she must remember the key. It is too cumbrous. Another difficulty, when the user is working on clear text mode on cryptographic editor, everyone can read the original text readily. Saving the encrypted file gradually is an approach but it is not possible moment by moment. Software should store the file on the hard disk and in some cases, storing it, on file servers by using a network. So, this action is not a secure idea because any intruder can listen through the network .Also, professional hackers may use of applications to read or rewrite the file easily.

So what is the way to avoid from user level holes? System level cryptography can improve some disadvantages of user level. Every time we are concern about our files. Sometimes they are on a physical media which is protected and sometimes they are on a file server. Users have to trust file server but in real situation any professional eavesdropper may stole data, especially when it is discussing in remote file server. By embedding an encryption algorithm on disks controller and considering a key, protecting the data also can be done on entire disks or every file blocks. So in this method, users must trust together and share a same key. Clearly, many users do not prefer to employ a same key. Another disadvantage of system level cryptography is related to the backups. Suppose user has made a backup without any encryption or with encryption. If the disk controller is not available, it cannot be useful to retrieve its sensitive data. Also, the backup needs to encrypt by a special encryption algorithm and a key. Therefore, implementing a cryptographic technique by using disk controller is not reliable and enough, especially to preserve files in remote file servers. Even though we ignore the disadvantages of

DOI: 10.7763/IJIEE.2013.V3.382

combining both of hardware (disk) encryption and cryptographic authentication, again this technique will encountered with using and maintaining key because each file access needs two operations with server, first, for the hardware (disk) and second, for the network. "Such a design violates the principle that work should be shifted from the (shared, heavily loaded) file server to the (unshared, lightly loaded) client machine whenever possible [4]".

Some operating systems on personal computer such as Macintosh use file encryption which is able to encrypt a specific area on the disk. Unfortunately, not only they depend on a particular type of storage (e.g. local hard drive) but also they usually need pre-allocation of disk space. Also, managing encrypted files are very crucial issue because for other systems, encrypted files show as a single large file. It is impossible to use conventional tools to handle these types of files. So, administers need new generation of tools for some general activities such as copy, move and so on.

III. Cryptography as an OS Service with CFS

As a basic question that is who is responsible for encryption? Or maybe it is better to say where is in charge of encryption and decryption? According to the previous section if we try to operate encryption in user level or system level the security of the system is not complete. Namely, in user level, user is responsible for encryption and it means that it may be with human's faults inadvertently. Also, we mentioned about the system level and its liabilities. So, cryptographic file system or CFS has a significant role to improve previous pitfalls. Besides of improving previous pitfalls, it gives peace of mind for every mistakenly user's errors. CFS has a crucial principle which is if each component is trustable it must be encrypted urgently before any communication to unreliable components.

IV. How CFS Can Help Us?

By settling CFS between system (hardware, low) level and user (high) level, improving all the previous deficiencies is possible. What advantages will get from implementing CFS? CFS's benefits as follows:

- Without using CFS all the users have to manage keys for every session. So, handling keys for every session is very difficult because each key after authenticating by system is not serviceable in next session. CFS can produce keys for each user in any session.
- Each encrypted files does not have any difference with others, except every encrypted files are useable with a special key. This definition is called transparency.
- In reality, encryption and decryption take extra time to perform them. But in CFS it is not so high. Especially, interactive requests do not differ from other types of responses.
- CFS protects file contents against differentiating between tow encrypted files. For instance, it is not possible for anyone to determine a specific pattern in two files in order to answer how files have been encrypted.
- Distributing file systems is very crucial issue. CFS can protect network connections when the file system should be use by networks [5].
- There is no difference between encrypted files and others in storing and managing. On the other hand, administrator does not require specific tools to backup and restore encrypted files. Also, administrators do not need to know about the keys when they would like to alter some parts.
- CFS support wherever the key will support. In terms of, portability is a crucial feature of it.
- CFS is not included overhead. Namely, when encryption is needed, file servers do not require any specific extra operation for clients who ask cryptographic processes.
- There is no constraint for file sharing. Therefore many users or processes can use encrypted files in a same time concurrently.
- CFS can be compatible with future advancements. For example, with advent of new types of cryptographic algorithm or somewhat devices such as smart card, it can be adjust with them because CFS is proportion of operating system.

V. CFS in Reality

In CFS's goal both of presenting the user with a cryptographic method in order to meet a secure file system in an integrated manner without difference between encrypted and common files, and also relieving the user that he or she does not need to enter or memorize the key for every session.

CFS also is empowered to encrypt directories. In this case, the user must issue a command in order to attach a key. When anyone tries to use these directories they must enter a key to see all the files and sub-directories. Not only there is no diversity in relate to the encrypted and usual directories for common administrative commands like backup and restore but also all of these operation are performed without key. CFS's dealing was various with files. As mentioned before, CFS encrypts the files when the user decides to write and decrypt them in reading time automatically.

In UNIX operating system, CFS provides a "virtual" file system for every client, usually it is mounted on /crypt. Both of files and file's path names are stored in encrypted form. All the directories which are encrypted can be available by remote file servers like Sun NFS [6] or even AFS [7]. CFS does not need a pre-allocation space to encrypt directories. On the other hand, user controls CFS by tools.

The keys in order to encrypt directories are produced via two ways. First, enter the keys character by character by user. For the purpose of generating the key for encryption, after entering the key by user via the keyboard, key is transformed to arbitrary-length which is utilized to create internal cryptographic keys. The length of arbitrary characters must be at least 16 characters for the purpose of several self-sufficient keys. Also, characters can be each useable ASCII code. Second, utilizing "smart card" has been connected to the computer. After entering the card's password by user, the key is copied from the card interface into the machine.

VI. Security

While authentication process ensure servers whether the user, clients or even services are trustable or not; file encryption or better to mention cryptographic file system (CFS) assures the clients, no one can access to the files without right key. On the other hand, file system cryptography guarantees authorization. Namely, if the user or others pass the authentication process after that they must be authorized by entering the right key in order to access and see the file's contents. Also CFS insures that files are never broadcasted in clear form across the networks. This is the most significant feature of the file encryption. Besides, it supports end to end encryption. Servers and its clients communicate altogether without any factual encryption at server side. Because of all the communications are performing on CFS mode server can trust to what it is storing on storage devices. Reciprocally, clients trust to all data which is sending from server in any time since CFS is running on the system.

Even though CFS protects files, directories, and communications between clients and servers but some security's aspects such as access time and file size have not been considered (CFS also encrypt all the symbolic link). On the other hand, if any intruder looks at file's access times then he asks from himself does this file is most important? Do many clients use this file? Similarly, file size also can diminish the final set of aims to find the most important files. Namely, intruder assumes a scenario to discover which file is confidential among all the files which have been encrypted by CFS. For instance, intruder looking for a letter relate to him from his manager. He knows the secretary types the letter in Microsoft Word. Therefore, he guesses a letter with 6 or 10 lines is not more than 10 KB. This is an idea to find files faster among cryptographic file system.

Despite the fact that, CFS known as a trustable cryptographic file system but it faces to this vulnerability, unfortunately. Especially, traffic analysis demonstrates this liability in both of making snapshots from files and real time observation. Even though, CFS has this difficulty but it must be mentioned which CFS has been designed to protect data only in basis of file system. It also should be considered which CFS is not a full general purpose cryptographic file system.

Another aspect of security in CFS is returning bits to user program. In other words, CFS's ability cannot protect it. It means that our information may be written when software is swapped out to a paging peripheral hardware. Particularly, when the paging devices are running on remote file system, the vulnerability goes up. Does CFS run on system as a paging file system? Theoretically, CFS also can perform paging file system, although it has not been implemented as a stable paging file system in real environment until now.

All the directories are checked by UNIX file protection which they are created under /crypt command. As we mentioned before, to show the file in form of clear text the user who has issued the cattach command can see it. Noteworthy, this feature is implemented by UID of each user. So, what happen when an attacker can access to a client illegally? Because we know when anyone login to the machine he or she can see the entire encrypted directory's name. By obscuring the file name after performing /crypt command nobody will able to see the directory's name in /crypt's list. Unfortunately, if attacker becomes "super user ", he or she will able to perform and modify information in anywhere in any form.

The system's security has a straight relation with encryption keys and level of attackers in order to make a successful prediction. Although brute force attack is enough for breaking any key but applying for all the directories, sub-directories, and files is not feasible in a reasonable time. For the purpose of increasing key's complexity UNIX check the length and entropy. Poorly selected key provide an easy guess for any attacker particularly when key is chosen by human, straightly. This type of attack is called dictionary-based attack. The primary action to frustrate dictionary-based attack is to choose a key which is adequately long. One of a cardinal point of the cmkdir is modifying the passphrase simply by enforcing user to choose a passphrase with high security such as character diversity and length. Unthinking selection and social engineering are very popular risks in relation with passphrase-based keys. Implementing some controls on user's activities and training can improve these types of risks properly.

In smart card based there is another story. Generating and storing the real encryption keys are done inside the smart cards. On the other hand, users only must memorize and preserve one passphrase which is related to access to use of the smart card. In order to start a file encryption, CFS fetches the keys from the smart cards and uses them to perform an encryption. Even though designing a process in which the keys will not need to leave the card theoretically but the smart card's bandwidth is not enough to operate entire cryptographic file system abilities such as file encryption or decryption inside the card, unfortunately.

VII. File Encryption

In order to encrypt files, CFS utilizes DES [8]. As you know, the diversity of DES's standard modes is plenty but none of them is entirely proper for encrypting online files. ECB or electronic code book is the easy and simple form of DES. In ECB each eight byte block of every file must be encrypted by a unique key, separately. Both of encrypting and decrypting can be done haphazardly on each block. Even though, encrypting each block is very striking but it unmasks a paradigm among of file about the file's structure. Namely, any professional intruder can differentiate between the files whether they have been encrypted or not. Nevertheless, this feature contributes a high protection for each unit of data.

Regretfully, because of DES uses long iterations it is not appropriate to implement on file system. Namely, more iteration has negative effect on file system operations.

VIII. Conclusion

Providing an easy and fundamental mechanism to preserve files and information which have been stored on disks or even exchanging data among files servers, need to develop and evolve CFS. Encryption by users (as a traditional perspective) is inappropriate. Producing more and more keys

in order to encrypt is a big trouble for any user. Also, improperly managing and storing keys prepare a potential threat for any attacker. Using more and more Cryptographic file systems can improve some liabilities and promote them for the purpose of establishing an easy, reliable, secure and fast environment.

REFERENCES

[1] D. Naccache and D. M'Raihi, "Cryptographic smart cards," *Micro, IEEE*, vol. 16, no. 3, pp. 14, 16-24, Jun. 1996.

[2] X. Lai and J. Massey, "A Proposal for a New Block Encryption Standard," in *Proc. EUROCRYPT 90*, pp. 389-404, 1990.

[3] National Bureau of Standards, "Data Encryption Standard," FIPS Publication #46, NTIS, Apr. 1977.

[4] R. K. Pal and I. Sengupta, "Enhancing file data security in linux operating system by integrating secure file system," *Computational Intelligence in Cyber Security, 2009. CICS '09. IEEE Symposium on*, pp. 45-52, March 30 2009-April 2 2009.

[5] National Bureau of Standards, *Data Encryption Standard Modes of Operation*, FIPS Publication #81, NTIS, Dec. 1980.

[6] J. H. Howard *et al.*, "Scale and Performanee in Distributed File Systems," *ACM Trans. Computing Systems*, vol. 6, no. 1, February, 1988.

[7] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network File System," in *Proc. USENIX*, 1985.

[8] R. Pletka and C. Cachin, "Cryptographic Security for a High-Performance Distributed File System," presented at Mass Storage Systems and Technologies, 24th IEEE Conference on, MSST 2007, pp. 227-232, 24-27 Sept. 2007.

Mohammad Reza Abbasy is a lecturer at Department of Computer Science, Tehran, Iran, Shamsipour Institute of Technology. His researches focus on Cryptography, Steganography, DNA Computing and Security Standards. He received his M.S. degree in Information Security from UTM (Malaysia).

Mahdi Sharifi is a PhD candidate in computer science at Universiti Teknologi Malaysia (UTM) and faculty member at department of computer engineering, Najafabad Branch, Islamic Azad University (IAU). His researches focus on trustworthy service computing, secure and trustable cloud services, security issues on Service Oriented Architecture (SOA), and computer network security. He received his B.S degree in computer engineering from IAU (Iran) and M.S degree in Information security from UTM.

Mohammadreza Najaftorkaman received a B.S. degree in Software Engineering from Shomal University in 2008 and a Master in information security from University Technology Malaysia in 2011. His research interests include information security, cryptography, steganography, and DNA cryptography.