ISSN: 2321-7782 (Online)
Volume 3, Issue 3, March 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

A Preliminary Review on File Protection Mechanism

Nilofar N. Pathan[1]
Computer Science and Engineering
H.V.P.M's C.O.E.T
Amravati, India

Ranjit R. Keole[2]
Professor
Information Technology
H.V.P.M's C.O.E.T
Amravati, India

Abstract: The Protection of a file is mostly needed in multi-user environment where a file is shared among several users. Nowadays, storage systems are increasingly subject to attacks. So the security system is quickly becoming mandatory feature of the data storage systems. For the security purpose we are always dependent on the cryptography techniques. These techniques take the performance costs for the complete system. So we have proposed the File Security System (FSS). It is based on the on-demand computing system concept, because of the performance issues. It is a greater comeback for the system performance. The concept is used because; we are not always in need the secure the files, but the selected one only.

Keywords: File protection mechanism, password, user-ID, File Encryption, Information Security.

I. Introduction

a) File System

File systems are complex programs designed for storing data on persistent storage devices such as disks. A file system manages the space available on the storage devices, provides the abstraction of files, which are data containers that can grow or shrink and have a name and other metadata associated to them, and manages the files by organizing them into a hierarchical directory structure. Internally, most file systems distinguish at least the following five components as shown in Figure.

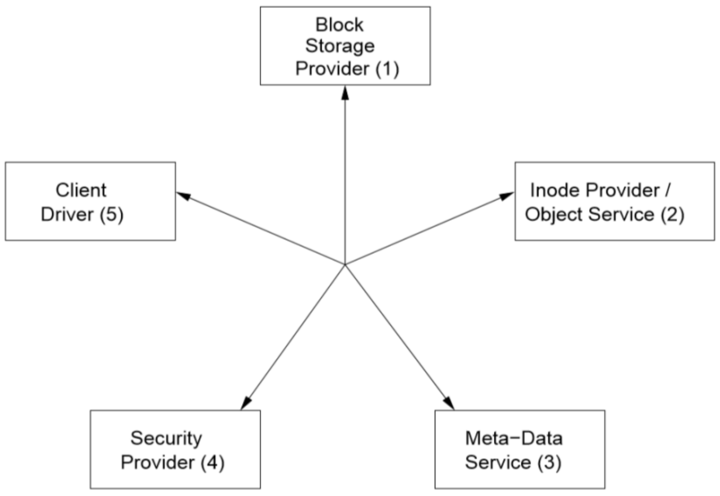

Block Storage Provider (1)
Client Driver (5)
Inode Provider / Object Service (2)
Security Provider (4)
Meta−Data Service (3)

Fig: Components of file system

1. *Block storage provider:* that serves as a bulk data store and operates only on fixed-size blocks.
2. *inode provider:* (or *object-storage service*), which provides a flat space of storage containers of variable size.
3. *Meta-data service:* handling abstractions such as directories and file attributes and coordinating concurrent data access.

4. *Security provider:* responsible for security and access-control Features.
5. *Client driver:* that uses all other components to realize the file system abstraction to the operating system on the client machine.

The first three components correspond to the layered design of typical file systems, i.e., data written to disk in a file system traverses the file-system layer, the object layer, and the block layer in that order. The security provider is usually needed by all three layers. In most modern operating systems, the block-storage provider is implemented as a block device in the operating system, and therefore not part of the file system.

File is one of the important resources of the computer system. That must be protected from the unauthorized access that it can't be tempered or stolen by intruders. The problem of File security does not only come from hacker Intrusion, Trojan horse programs and other security threats from the internet. In fact more security threats come from enterprise itself because of lack of security experience. A lot of important documents are stored in a sensitive position directly, such as desktop and my documents folder in windows operating system. The file security can enforced using cryptographic techniques. With the help of these techniques the important files are encrypted and authorized users are given appropriate cryptographic keys.

Security is quickly becoming a mandatory feature of data or file storage systems. Today, storage space is typically provided by complex networked systems. These networks have traditionally been confined to data centers in physically secured locations. But with the availability of highspeed LANs and storage networking protocols such as FCIP and iSCSI , these networks are becoming virtualized and open to access from user machines. Hence, clients may access the storage devices directly, and the existing static security methods no longer make sense. New, dynamic security mechanisms are required for protecting stored data in virtualized and networked storage systems. A secure storage system should protect the confidentiality and the integrity of the stored data.

II. Literature Review And Related Work

A considerable number of prototype and production cryptographic file systems have been developed in the past 15 years. So many approaches are applied to solve the problem of information security. The approaches may be the user space or kernel space or the combined one. The kernel approach is sensitive to implement because any small mistake done by the programmer can harm the overall functioning of the system. The user space one is secure and compatible with the system and the independent one and comfortable in the implementation and are the highly portable if we are using the best portable platform like Java. Most early cryptographic file systems are layered and use the NFS protocol for accessing a lower-layer file system: CFS uses an NFS loopback server in user space and provides per-directory keys that are derived from passphrases; TCFS uses a modified NFS client in the kernel and utilizes a hierarchical key management scheme, in which per-user master keys to protect per-file keys are maintained. SFS is a distributed cryptographic file system also using the NFS interfaces, which is available for several Unix variants. These systems do not contain an explicit security provider responsible for key management, and delegate much of that work to the user.

A cryptographic file system has also been implemented using secure network-attached disks (SNAD) . SNAD storage devices are a hybrid design, providing traditional block storage as well as features commonly found in object-storage devices and file servers. In contrast to traditional storage providers, SNAD devices require strong client authentication for any operation and also perform data verification on the content. Data is encrypted by the clients before sending it to the SNAD and authenticated using per-block digital signatures or per-block secret-key authentication (MAC). Sand-box security model is extremely useful for secure execution of untrusted applications; many security systems based on sand-box model so far provide security by intercepting system calls invoked by applications and controlling their execution.

III. Basic Categories of File Protection Mechanism

The system has attributes as follows: Firstly, it does dynamic encryption or decryption for sensitive data at driver level and the operators needn't get involved in. Secondly, the system can be used in LAN sharing and it ensures the safety of data which is transmitted from one operator to another. At last, the system can automatically record the user's computer information and store them in the file which is operated. If the file was leaked, we can trace the transmission path and find who leaked it according to the information recorded before. More importantly, this novel system has been tested in real network and the results show the efficiency and convenience of our design.

As sensitive data in personal computers are facing more and more security threats, we design a traceable file protection system with transparent encryption technology in this paper. It combines prevention with diagnosis of data leakage and achieves multi-level protection of data. All multiuser operating systems must provide some minimal protection to keep one user from maliciously or inadvertently accessing or modifying the files of another. As the number of users has grown, so also has the complexity of these protection schemes.

a) All "None Protection

Files were by default public, in the original IBM OS operating systems,. Any user could read, modify, or delete a file belonging to any other user. Instead of software- or hardware-based protection, the principal protection involved trust combined with ignorance. System designers supposed that users could be trusted not to read or modify others' files, because the users would expect the same respect from others. It was acknowledged that certain system files were sensitive and that the system administrator could protect them with a password. A normal user could exercise this feature, but passwords were viewed as most valuable for protecting operating system files. Two philosophies guided password use. Sometimes, passwords were used to control all accesses (read, write, or delete), giving the system administrator complete control over all files. But at other times passwords would control only write and delete accesses, because only these two actions affected other users. In either case, the password mechanism required a system operator's intervention each time access to the file began.

b) Group Protection

Because the all-or-nothing approach has so many drawbacks, researchers focused on identifying groups of users who had some common relationship. In a typical implementation, the world is divided into three classes: the user, a trusted working group associated with the user, and the rest of the users. For simplicity we can call these classes user, group, and world . All authorized users are separated into groups. A group may consist of several members working on a common project, a department, a class, or a single user. The basis for group membership is needed to share. The group members have some common interest and therefore are assumed to have files to share with the other group members. In this approach, no user belongs to more than one group. When creating a file, a user defines access rights to the file for the user, for other members of the same group, and for all other users in general. Typically, the choices for access rights are a limited set, such as {read, write, execute, delete}. A key advantage of the group protection approach is its ease of implementation. A user is recognized by two identifiers : a user ID and a group ID. These identifiers are stored in the file directory entry for each file and are obtained by the operating system when a user logs in. Therefore, the operating system can easily check whether a proposed access to a file is requested from someone whose group ID matches the group ID for the file to be accessed.

c) Single Permissions

In spite of their drawbacks, the file protection schemes we have described are relatively simple and straightforward. The simplicity of implementing them suggests other easy-to-manage methods that provide finer degrees of security while associating permission with a single file.

d) Password or Other Token

We can apply a simplified form of password protection to file protection by allowing a user to assign a password to a file. User accesses are limited to those who can supply the correct password at the time the file is opened. The password can be required for any access or only for modifications (write access).

e) Temporary Acquired Permission

The Unix operating system provides an interesting permission scheme based on a three-level user “group “world hierarchy. The Unix designers added a permission called set userid (suid) . If this protection is set for a file to be executed, the protection level is that of the file's owner , not the executor . To see how it works, suppose Tom owns a file and allows Ann to execute it with suid . When Ann executes the file, she has the protection rights of Tom, not of herself.

This peculiar-sounding permission has a useful application. It permits a user to establish data files to which access is allowed only through specified procedures. This mechanism is convenient for system functions that general users should be able to perform only in a prescribed way. For example, only the system should be able to modify the file of users' passwords, but individual users should be able to change their own passwords any time they wish. With the SUID feature, a password change program can be owned by the system, which will therefore have full access to the system password table. The program to change passwords also has SUID protection, so that when a normal user executes it, the program can modify the password file in a carefully constrained way on behalf of the user.

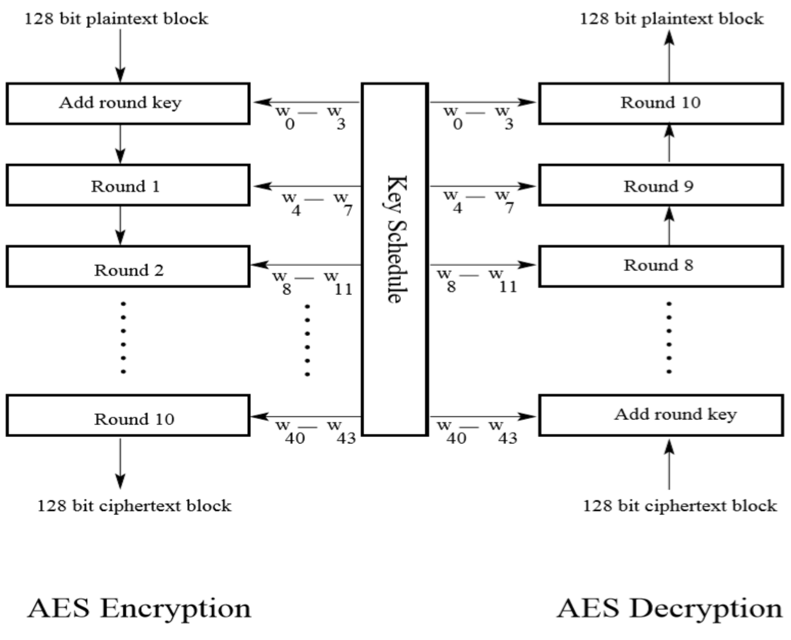
f) Per-Object and Per-User Protection

The primary limitation of these file protection schemes is the ability to create meaningful groups of related users who should have similar access to one or more data sets. The access control lists or access control matrices described earlier provide very flexible protection. Their disadvantage is for the user who wants to allow access to many users and to many different data sets; such a user must still specify each data set to be accessed by each user. As a new user is added, that user's special access rights must be specified by all appropriate users.

IV. Architecture of File Protection With Encryption

Every time a file on the data partition is created, Data Protection creates a new 256-bit key (the “per-file” key) and gives it to the hardware AES engine, which uses the key to encrypt the file as it is written to flash memory using AES CBC mode. The initialization vector (IV) is calculated with the block offset into the file, encrypted with the SHA-1 hash of the per-file key. The per-file key is wrapped with one of several class keys, depending on the circumstances under which the file should be accessible. Like all other wrappings, this is performed using NIST AES key wrapping, per RFC 3394. The wrapped per-file key is stored in the file’s metadata. When a file is opened, its metadata is decrypted with the file system key, revealing the wrapped per-file key and a notation on which class protects it. The per-file key is unwrapped with the class key, then supplied to the hardware AES engine, which decrypts the file as it is read from flash memory. The metadata of all files in the file system is encrypted with a random key, which is created when iOS is first installed or when the device is wiped by a user. The file system key is stored in Effaceable Storage. Since it’s stored on the device, this key is not used to maintain the confidentiality of data; instead, it’s designed to be quickly erased on demand (by the user, with the “Erase all content and settings” option, or by a user or administrator issuing a remote wipe command from a mobile device management (MDM) server, Exchange ActiveSync, or iCloud). Erasing the key in this manner renders all files cryptographically inaccessible.

The content of a file is encrypted with a per-file key, which is wrapped with a class key and stored in a file’s metadata, which is in turn encrypted with the file system key. The class key is protected with the hardware UID and, for some classes, the user’s passcode. This hierarchy provides both flexibility and performance. For example, changing a file’s class only requires rewrapping its per-file key, and a change of passcode just rewraps the class key.



V. Conclusion

We have achieved the high security by including the support of the Rijndeal Algorithm (AES) and we have saved the keys on the portable smart cards for the documents which are important. The performance is achieved with the help of on-demand computing concept which is that we are not going to encrypt all the files on the computer system, but we are going to encrypt only the important documents only. It saves the performance overhead of the system. In reviewing the extent to which protection mechanisms are systematically understood (which is not a large extent) and the current state of the art, one cannot help but draw a parallel between current protection inventions and the first mass produced computers of the 1950's. At that time, by virtue of experience and strongly developed intuition, designers had confidence that the architectures being designed was complete enough to be useful.

Acknowledgment

My thanks to the Guide, Prof. R.R.Keole and Principal Dr.A.B.Marathe, who provided me constructive and positive feedback during the preparation of this paper.

References

1. B. Kahanwal, T. P. Singh, and R. K. Tuteja, “Towards the Framework of the File Systems Performance Evaluation Techniques and the Taxonomy of Replay Traces”, International Journal of Advanced Research in Computer Science (2011), Vol. 2, No. 6, pp. 224-229..
2. A. Azagury, R. Canetti, M. Factor, S. Halevi, E. Henis, D. Naor, N. Rinetzky, O. Rodeh, and J. Satran, “A two layered approach for securing an object store network,” in Proc. 1st International IEEE Security in Storage Workshop (SISW 2002), 2002.
3. A. Azagury, V. Dreizin,M. Factor, E. Henis, D. Naor, N. Rinetzky, O. Rodeh, J. Satran, A. Tavory, and L. Yerushalmi, “Towards an object store,” in Proc. IEEE/NASA Conference on Mass Storage Systems and Technologies (MSST 2003), pp. 165–177, 200.
4. A. Westin, Privacy and Freedom. New York: Atheneum, 1967. (I-A1, SFR)
5. A. Miller, The Assault on Privacy. Ann Arbor, Mich.: Univ. of Mich. Press, 1971; also New York: Signet, 1972, Paperback W4934. (I-A1, SFR)
6. Dept. of Health, Education, and Welfare, Records, Computers, and the Rights of citizens. Cambridge, Mass.: M.I.T. Press, 1973. (I-A1, SFR)
7. R. Turn and W. Ware, "Privacy and security in computer systems," I-A1 Amer. Scientist, voL 63, pp. 196-203, Mar.-Apr. 1975. (I-A1)
8. W. Ware, "Security and privacy in computer systems," in 1967 S,CC, AFIPS Cont. Proc., vol. 30, pp. 287-290. (I-A1)

AUTHOR(S) PROFILE

Nilofar N. Pathan, received the B.E.degree in Information Technology from H.V.P.M's College Of Engineering And Technology, Amravati in 2014. She is currently persuing Master's Degree in Computer Science and Engineering from H.V.P.M's College of Engineering And Technology, Amravati.

Prof. Ranjit R.Keole, received the B.E.and M.E degree in Computer Science from Prof. Ram Meghe Institute of Technology, Badnera in 1992 and 2008, respectively. His field of specialisation is web Mining. He is currently working as Associate Professor at H.V.P.M's college of Engineering and Technology,Amravati.