Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 22(2): 121-126, Jun. 2024 Regular paper

Key Management Server Design in Multiuser Environment for Critical File Protection

Sung-Hwa Han[1]*

[1]Department of Information Security, TongMyong University, Busan, 48520, Korea

Abstract

In enterprise environments, file owners are often required to share critical files with other users, with encryption-based file delivery systems used to maintain confidentiality. However, important information might be leaked if the cryptokey used for encryption is exposed. To recover confidentiality, the file owner must then re-encrypt and redistribute the file along with its new encryption key, which requires considerable resources. To address this, we propose a key management server that minimizes the distribution of encryption keys when critical files are compromised, with unique encryption keys assigned for each registered user to access critical files. While providing the targeted functions, the server employs a level of system resources comparable to that of legacy digital rights management. Thus, when implemented in an enterprise environment, the proposed server minimizes cryptokey redistribution while maintaining accessibility to critical files in the event of an information breach.

Index Terms: Digital Right Management, Encryption Key, File Protection, Key Management System, Knowledge Management System

I. INTRODUCTION

As the social environment becomes more complex, companies and organizations must handle increasing volumes of critical information and media [1]. Much of this information, especially that used to predict future outcomes, is confidential [2]. Confidential information can only be accessed by a limited number of users, and may be shared among users to maximize operational efficiency [3].

Confidential information is typically safeguarded via access restrictions [4]. This information is prone to leaks or modifications if made available to unauthorized parties [5]. Specifically, malicious attackers may compromise an organization's activities by accessing important information [6]. Therefore, various technologies have been developed to prevent unauthorized access to such information. Confidential information can be protected by denying access to unauthorized users via access control, or maintaining confidentiality using encryption [7].

When critical files are transferred externally, it becomes difficult to enforce access control. Therefore, systems that rely on external transfer typically maintain confidentiality using encryption techniques [8]. Many organizations use open-source software (OSS)-based file encryption methods or commercial digital rights management (DRM) technologies [9]. When an encrypted file is shared, it is transferred to another channel along with its cryptokey, and subsequently sent to the recipient. The cryptokey allows only authenticated users to access the file.

A DRM-based file sharing system stores cryptokeys before delivering them to authenticated users [10]. However, malicious parties are able to obtain the cryptokeys using social engineering attacks or malware. To maintain the confidentiality of a file, the file owner must re-encrypt the file, store

Received 19 September 2023, **Revised** 8 November 2023, **Accepted** 15 November 2023
***Corresponding Author** Sung-Hwa Han (E-mail: shhan@tu.ac.kr, Tel:+82-10-3943-1247)
Department of Information Security, TongMyong University, Busan 48520, Korea

Open Access https://doi.org/10.56977/jicce.2024.22.2.121 print ISSN: 2234-8255 online ISSN: 2234-8883

the new cryptokey in DRM, and redistribute the cryptokey to all authenticated users. Although this process reinforces the security of sensitive data, it is highly resource-intensive.

To mitigate this issue, we propose a key management server designed for multiuser environments.

The proposed key management server enables the following functions:

- When a user creates a critical file, an encryption key is randomly generated.
- When the file owner selects users to access the critical files, the cryptokey is encrypted with the corresponding users' keys.
- When a user attempts to access the file, their authority is verified through a user authentication process.
- Upon authentication, the encrypted cryptokey is delivered to the shared user.

The primary contributions of the proposed file access monitoring structure are as follows:

- Explicit access to each file can be managed by its owner, who can directly specify authorized users.
- Users can share critical files using unique cryptokeys.
- Cryptokeys can be readily redistributed even following exposure.

The proposed key management server was demonstrated to outperform legacy DRM in terms of functionality, and its effectiveness was verified by analyzing its functionality and performance.

II. RELATED WORKS

A. File protection techniques

Organizations frequently use information –including text, images, audio, video, and floor plans– that is stored in files and shared with other users [11] via methods such as knowledge management systems (KMS) and web-based boards [12]. Accordingly, network file systems typically allow users to share files online [13]. When such files contain confidential or sensitive information, it is critical to maintain a sufficient level of confidentiality via access control.

Access control is provided by multiple functionalities including umask, SELinux, AppArmor, and Group Policy Objects [14].

Fig. 1 depicts the architecture of SELinux, which monitors file access by users [15]. When a user attempts to access a file, the file is compared with the access control policy registered on the security server. If the user is not authorized to access the file, access is denied at the kernel level in the access vector cache.

In web-based file sharing systems such as KMS, access control functions can be provided by enforcing user identification and authentication. By providing account and privilege management functions, web services regulate access to critical files according to user identity or level of information [16].

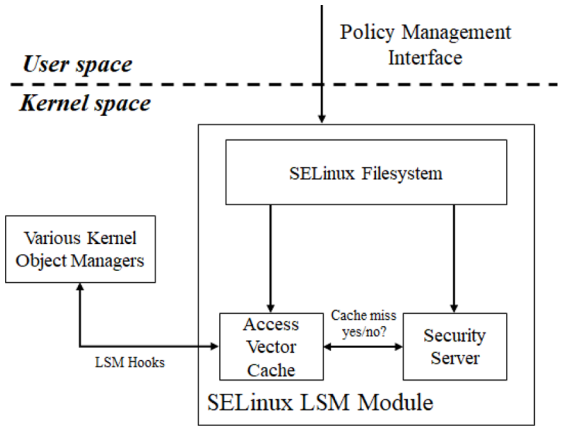

Fig. 1. SELinux file access control architecture

The confidentiality of files during transfer is typically maintained via encryption [17]. In an encryption-based file delivery system, files are initially encrypted by their owners, and subsequently sent to authorized users through a different channel, along with the corresponding cryptokeys. An authenticated user can access each file after decrypting it using its respective cryptokey. However, this approach is ineffective because it requires file owners to directly manage encryption keys. This inefficiency can be mitigated using DRM.

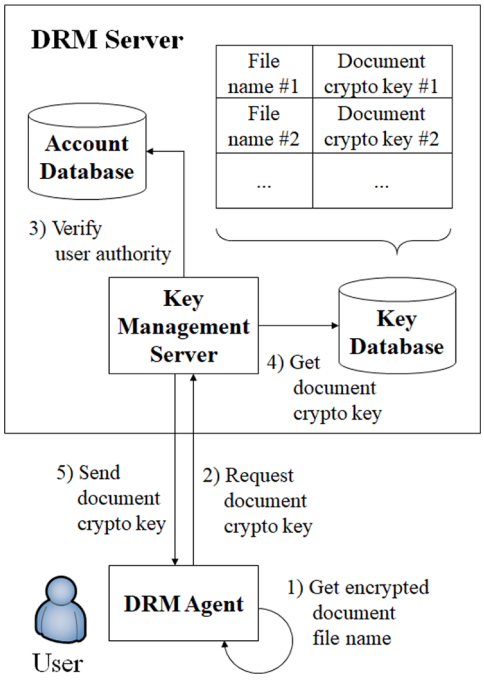

Fig. 2. DRM structure

Fig. 2 illustrates a DRM structure [18] comprising a server and an agent, where the former consists of a key management server, account database, and key database. Using the DRM agent, the user creates a cryptokey and uses it to encrypt critical files [19]. The generated cryptokey is delivered to the key management server, which manages the crypto-keys and provides user authentication functions.

All encrypted files, along with their respective cryptokeys, are stored in the key database. To access a file, the user must be authenticated by the key management server. Upon successful authentication, the server sends the corresponding cryptokey to the user.

B. Analysis of security environments

DRM-based file delivery systems use encryption techniques that allow multiple users to share files while maintaining confidentiality. Although OSS-based file sharing techniques are currently in development, they incur potential security vulnerabilities.

First, file sharing systems are vulnerable to damage from cryptokey leaks. In the process of sharing files, users also share the corresponding cryptokeys, which may be intercepted by attackers through packet sniffing or e-mail hooking. Upon accessing critical files, attackers may leak or modify the stored information.

Furthermore, cryptokey leaks incur high response costs, as file owners must re-encrypt any compromised files to maintain confidentiality. Because this process must be repeated to share the re-encrypted files with authorized users, it is highly resource-intensive.

III. KEY DATABASE DESIGN IN MULTIUSER ENVIRONMENT

To improve file transfer efficiency and expand the functionalities of legacy DRM, we propose a key management server in a multiuser environment.

A. Key database design

The proposed key management server simultaneously handles legacy DRM accounts and databases. For each critical file, the database records the IDs of all users authorized to access the file. Furthermore, the database includes an account list table, which associates cryptokeys with their respective users. Fig. 3 illustrates the key database structure.

The key management server separates each cryptokey used to encrypt a critical file from the user key that encrypts it. The cryptokey stored in the database is encrypted with the key of the user accessing the file. Thus, N encrypted cryptokeys are created for a system of N authorized users, with each user assigned a unique key. Even if a user changes their key, accessibility can be guaranteed by modifying only the encrypted cryptokey. In the event that a user key is compromised, the keys of all other users remain secure, and only the leaked user key must be regenerated and redelivered. This is more cost-effective than the key redistribution process used in legacy DRM.

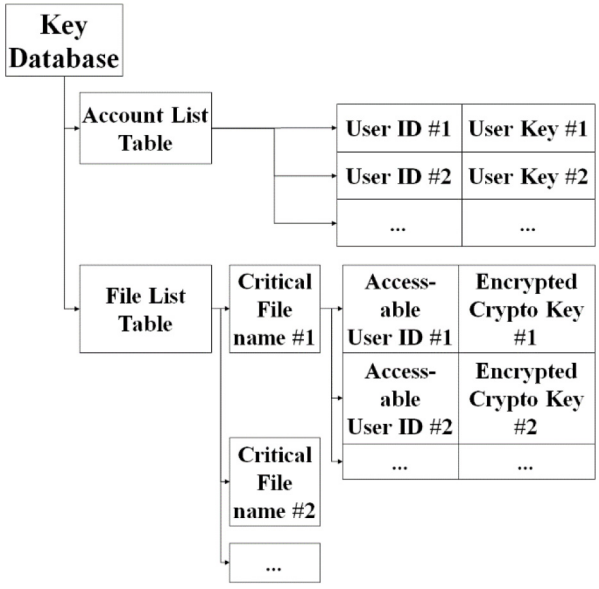

Fig. 3. Key table structure

B. Critical file encrypt/decrypt process

Fig. 4 illustrates the encryption and decryption of files by the proposed key management server.

When the DRM manager registers a user account, the server stores a corresponding user key. The DRM agent randomly generates a cryptokey and encrypts the user-generated file. Subsequently, the agent delivers the cryptokey, filename, and an authorized user list defined by the file owner to the server. The server then generates an encrypted cryptokey with each user key in the list of authorized users and stores it in the key database.

When requesting access to a file, the user first sends their ID to the server through the DRM agent and requests an encrypted cryptokey. The server searches the database and verifies whether the user is registered in the authorized user list. Upon successful authentication, the server delivers the encrypted cryptokey to the user.

By entering the received key, the user can decrypt and access the desired file.

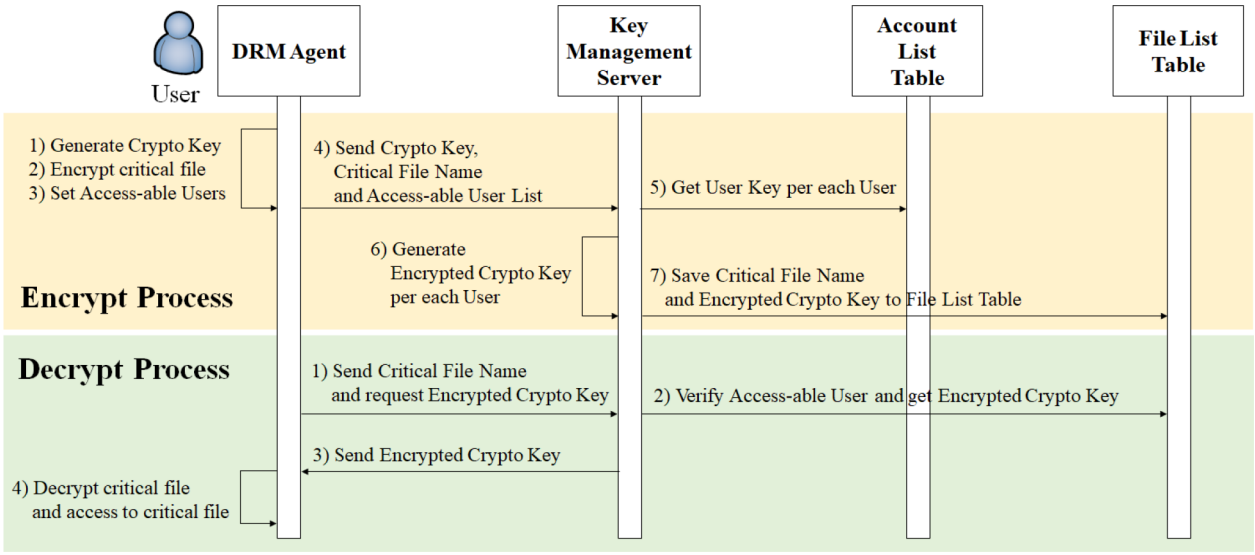

Fig. 4. File encryption and decryption processes

IV. VERIFICATION

A. Verification environment

By resolving the functional problems associated with legacy DRM, the proposed key management server can accurately provide targeted functions without incurring additional time or resource costs. The effectiveness of the proposed server was verified by analyzing its functionality and performance.

Table 1 lists the software specifications and hardware environment used to verify the proposed server. The server and key databases were implemented in a separate system from the DRM agent. An artificially low hardware was selected to measure the resource usage of the proposed server and DRM agent.

Table 1. Verification environments

Component	Key Management Server and Database	DRM Agent
OS	CentOS 8.3	CentOS 8.3
CPU	Intel i5 3750K	Intel i5 3750K
Memory	8 Gbyte	8 Gbyte
SSD	500Gbyte	500Gbyte

B. Function verification

The proposed key management server must be able to regulate access by multiple users while maintaining the confidentiality of critical files. Any authenticated users must be simultaneously granted access to the files via user keys, whereas unauthenticated users should be denied access. The defined unit functions are listed in Table 2.

Table 2. Function Verification Items

ID	Object
Func_01	When registering a user account, check whether the unique user key is created and saved properly.
Func_02	When a user creates a critical file, check whether the cryptokey required to encrypt it is generated properly.
Func_03	With the owner's settings, check whether the critical file is encrypted with the cryptokey.
Func_04	When the owner specifies users authorized to access the critical file, check whether the encrypted cryptokey is generated using the key of each user registered in the authorized user list.
Func_05	When multiple users access a critical file, check whether the encrypted cryptokeys are decrypted using corresponding user keys. Subsequently, check whether the file is decrypted using the decrypted cryptokeys.
Func_06	When an unauthorized user gains access to a critical file, the key management server identifies this user by searching the key database.

To confirm that the proposed server provides the intended functionality, we executed the verification items listed in Table 2. Because the actual results were similar to the expected results, the proposed key management server has been demonstrated to overcome the functional limitations of legacy DRM.

C. Performance verification

To verify the proposed server's performance in a multi-user environment, we conducted a performance test using OpenSSL 1.1.1v, with a legacy file encryption technique adopted as a baseline.

1) Cryptokey input resource usage

In an environment where multiple users can access critical files, the resources used to store encrypted cryptokeys should minimize interference with the activities of other applications. Therefore, the resource usage of the proposed server was measured and compared to that of OpenSSL 1.1.1v. To this end, we measured CPU usage when generating 10000, 25000, 50000, 100000, 250000, and 500000 encrypted cryptokeys. Each configuration was deployed and measured 10 times, with Table 3 listing the average measured CPU usage for each set of trials [20].

As observed from the table, the two architectures incurred similar resource costs to generate cryptokeys.

Table 3. Resource usage when generating cryptokeys

Case		Crypto Key Count		
Key Database suggest in this study	Count	10000	25000	50000
	CPU Usage	1.22%	1.24%	1.23%
	Count	100000	250000	500000
	CPU Usage	1.20%	1.23%	1.22%
OpenSSL 1.1.1v	Count	10000	25000	50000
	CPU Usage	1.20%	1.25%	1.22%
	Count	100000	250000	500000
	CPU Usage	1.21%	1.22%	1.21%

2) Cryptokey access time

To measure the time required to access cryptokeys by the proposed server and OpenSSL, we registered 10000, 25000, 50000, 100000, 250000, and 500000 dummy cryptokeys and measured the cryptokey search time as follows:

Encrypted crypto key search time = Time to read the final encrypted crypto key – Time to read the first encrypted crypto key in the table.

As shown in Fig. 5, the search times measured for the two architectures were mutually comparable.

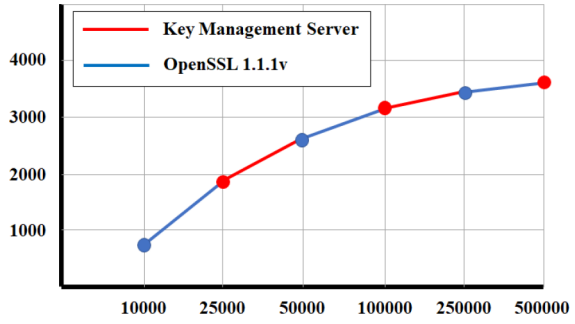

Fig. 5. Time required to search for encrypted cryptokeys

V. CONCLUSION

As enterprise environments grow in complexity, organizations and companies must handle increasingly large volumes of information, and real-time responsiveness must be maintained in the process of sharing this information. In this environment, data confidentiality can be maintained using encryption-based DRM. However, this approach is vulnerable to cryptokey leaks, which require considerable resources to recover from.

To address this problem, we propose a key management server structure that minimizes re-encryption costs in a multi-user environment by assigning a unique key to each authorized user. When a user key is lost or exposed, the confidentiality of shared data can be maintained by modifying only that specific key.

The functionality and performance of the proposed server were verified to be effective. Our experimental results confirm that the target functionality is provided accurately, with resource usage and response speeds comparable to those of OpenSSL 1.1.1v.

However, the present study focused solely on presenting and verifying the core functionality of the proposed server. Therefore, additional research is required to implement the server in real enterprise environments.

ACKNOWLEDGMENTS

This Research was supported by Tongmyong University Research Grant 2021A023.

REFERENCES

[1] S. H. Han, “TTY Session Audit Techniques for Linux Platform,” in *IEEE/ACIS International Conference on Big Data, Cloud Computing, and Data Science Engineering, Cham: Springer International Publishing*, pp. 95-105, Feb. 2023. DOI: https://doi.org/10.1007/978-3-031-19608-9_8.

[2] R. Wądołowski, “Protection of classified information in Bosnia and Herzegovina and Croatia,” *Selected criminal and administrative regulations. Przegląd Bezpieczeństwa Wewnętrznego*, vol. 14, no. 27, pp. 276-299, Dec. 2022. DOI: https://doi.org/10.4467/20801335PBW.22.059.16950.

[3] V. S. Tchamyou, “The role of information sharing in modulating the effect of financial access on inequality,” *Journal of African Business*, vol. 20, no. 3, pp. 317-338, Mar. 2019. DOI: https://doi.org/10.1080/15228916.2019.1584262.

[4] J. L. Peterson, “Confidentiality in medicine: how far should doctors prioritise the confidentiality of the individual they are treating?,” *Postgraduate medical journal*, vol. 94, no. 1116, pp. 596-600, Oct.

2018. DOI: https://doi.org/10.1136/postgradmedj-2018-136038.
[5] P. Yang, N. Xiong, and J. Ren, "Data security and privacy protection for cloud storage: A survey," *IEEE Access*, vol. 8, pp. 131723-131740, Jul. 2020. DOI: https://doi.org/10.1109/ACCESS.2020.3009876.
[6] G. R. Tsochev, R. D. Yoshinov, and O. P. Iliev, "Key problems of the critical information infrastructure through SCADA systems research," *Информатика и автоматизация*, vol. 18, no. 6, pp. 1333-1356, Dec. 2019. DOI: https://doi.org/10.15622/sp.2019.18.6.1333-1356.
[7] Z. N. Mohammad, F. Farha, A. O. Abuassba, S. Yang, and F. Zhou, "Access control and authorization in smart homes: A survey," *Tsinghua Science and Technology*, vol. 26, no. 6, pp. 906-917, Jun. 2021. DOI: https://doi.org/10.26599/TST.2021.9010001.
[8] M. P. K. Bachhav and M. M. A. Amritkar, "Secure Data Access Control and Efficient CP-ABE for Multi Authority Cloud Storage with Data Mirroring," in *International Conference On Emanations in Modern Technology and Engineering*, vol. 5, no. 3, pp. 19-22, 2017.
[9] H. E. R. Hassan, M. Tahoun, and G. S. ElTaweel, "A robust computational DRM framework for protecting multimedia contents using AES and ECC," *Alexandria Engineering Journal*, vol. 59, no. 3, pp. 1275-1286, Jun. 2020. DOI: https://doi.org/10.1016/j.aej.2020.02.020.
[10] C. C. Lee, C. T. Li, Z. W. Chen, Y. M. Lai, and J. C. Shieh, "An improved E-DRM scheme for mobile environments," *Journal of information security and applications*, vol. 39, pp. 19-30, Apr. 2018. DOI: https://doi.org/10.1016/j.jisa.2018.02.001.
[11] P. Kaushik, K. Joshi, J. Pandey, and T. Garg, "Statistical Deformity in Steganogrpahy and its Overcomings," *Journal of Emerging Technologies and Innovative Research*, vol 5, no. 6, pp. 697-710, Jun. 2018.
[12] E. Sultanow, M. Tobolla, A. Ullrich, and G. Vladova, "Visual Analytics Supporting Knowledge Management," in *i-KNOW*, Oct. 2017.
[13] K. Matsuzawa, M. Hayasaka, and T. Shinagawa, "The quick migration of file servers," in *Proceedings of the 11th ACM International Systems and Storage Conference*, pp. 65-75, 2018. DOI: https://doi.org/10.1145/3211890.3211894.
[14] H. K. Lee, S. H. Han, and D. Lee, "Kernel-Based Container File Access Control Architecture to Protect Important Application Information," *Electronics*, vol. 12, no. 1, pp. 52, Dec. 2022. DOI: https://doi.org/10.3390/electronics12010052.
[15] S. H. Han and D. Lee, "Kernel-based real-time file access monitoring structure for detecting malware activity," *Electronics*, vol. 11, no. 12, pp. 1871, Apr. 2022. DOI: https://doi.org/10.3390/electronics11121871.
[16] S. Cho, S. Hwang, W. Shin, N. Kim, and H. P. In, "Design of military service framework for enabling migration to military SaaS cloud environment," *Electronics*, vol. 10, no. 5, pp. 572, Mar. 2021. DOI: https://doi.org/10.3390/electronics10050572.
[17] L. Yang, Z. Han, Z. Huang, and J. Ma, "A remotely keyed file encryption scheme under mobile cloud computing," *Journal of Network and Computer Applications*, vol. 106, pp. 90-99, Mar. 2018. DOI: https://doi.org/10.1016/j.jnca.2017.12.017.
[18] H. Wang, "A Password-Based Access Control Framework for Time-Sequence Aware Media Cloudization," *Cryptology ePrint Archive*, Oct. 2022.
[19] D. Cho, S. Hwang, and G. Jeong, "DRM market system for media service platform supporting multi-DRM in cloud environment," *Advanced Science Letters*, vol. 23, no. 12, pp. 12721-12724, Dec. 2017. DOI: https://doi.org/10.1166/asl.2017.10886.
[20] A. E. Guvercin and B. Avenoglu, "Performance Analysis of Object-Relational Mapping (ORM) Tools in. Net 6 Environment," *Bilişim Teknolojileri Dergisi*, vol. 15, no. 4, pp. 453-465, Oct. 2022. DOI: https://doi.org/10.17671/gazibtd.1059516.

Sung-Hwa Han
He received the Ph.D. degree from Soongsil University, Republic of Korea. He is a professor in the Department of Information Security, Tongmyong University. He has been developing information security techniques for about 20 years. His research areas inlcude network, system security, and various platform security, such as cloud platform and blockchain.