Study of Transparent File Encryption Technology on Android Platform

Yongzhong Li(✉), Shipeng Zhang, and Yi Li

School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212003, China
liyongzhong6l@163.com

Abstract. Aiming at the data security problem of Android platform, a transparent encryption system based on file filter driver is designed and implemented, according to the technology of file transparent encryption and decryption system based on hook transparent encryption technology and file filtering driven transparent encryption technology used on windows platform. This system is different from the traditional APP development method of Android system. By intercepting the system call function and using the secret-key converted from the host MAC address, the encryption and decryption algorithm is written into the kernel, which fundamentally guarantees the security of user information. At the same time, the user's security experience is improved by putting authentication on the screen unlocking. The system design and implementation are described in this paper from system requirement analysis to overall design and detailed design of each module. Android application development technology and cross-compiling principle are used in the coding process. The system test results show that the system can effectively transparently encrypt files and protect the privacy of mobile files.

Keywords: Android · Data security · Transparent encryption · Privacy preservation

1 Introduction

As an open source mobile development platform, Android has been supported by mobile phone users and developers all over the world. In May 2017, at the 2017Google Developers Conference, Google announced that the number of smartphones using Android has reached 2 billion, close to one third of the world's population. However, while people enjoy the convenience brought by mobile phones, as an important data carrier of daily life and work, their security problems are becoming increasingly prominent.

At present, file transparent encryption technology has become increasingly mature. However, it is mostly used in Windows platform, and the application market for Android mobile phone file encryption software is uneven, and users are required to enter passwords to verify every time they encrypt and decrypt files, which greatly reduce the encryption efficiency and user experience. A transparent encryption system based on Android file filter driver is designed in this paper. In the kernel layer, the

G. Wang et al. (Eds.): SpaCCS 2020, LNCS 12383, pp. 137–145, 2021.
https://doi.org/10.1007/978-3-030-68884-4_11

encryption and decryption algorithm are written into the kernel by intercepting system calls, so as to improve user experience and encryption efficiency. The system's authentication is placed in the screen lock.

2 Android System Architecture

Android system architecture is based on the Linux kernel and is bottom-up structure [1]. It is mainly divided into four layers, as shown in Fig. 1, the Linux Kernel layer, the Library layer, the Application Framework layers and the Application layer. The Linux kernel layer provides the underlying drivers for various hardware of Android devices, such as display driver, audio driver, etc. The system runtime layer mainly provides the main features support for Android system through some C/C++ libraries, such as SQLite library, etc. The application framework layer mainly provides various APIs that may be used to build applications. Application Programming Interface (API); the application layer includes all applications installed on mobile phones [2].

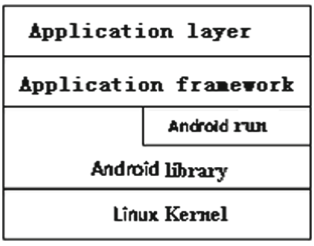

Fig. 1. Android system architecture

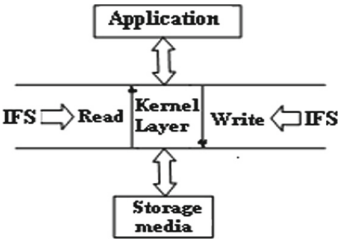

Fig. 2. Transparent encryption

3 Principle of Transparent Encryption Technology

Transparent encryption refers to the process of encrypting and decrypting files without changing the user's operating habits. It is a passive compulsory encryption technology [3], which is insensitive to users. When the user opens or edits the specified file, the system will automatically encrypt the unencrypted file and decrypt the encrypted file. Encrypted files leave the current usage environment, which can not automatically decrypt and protect the contents of files.

Transparent encryption technology can be divided into user-mode implementation and kernel-mode implementation according to the location of implementation. They correspond to the two main transparent encryption technologies, namely hook transparent encryption and file filter-driven transparent encryption. According to encryption efficiency, hook encryption technology encrypts the whole file in the application layer, and encrypts and decrypts the file relatively slowly. Driving transparency technology encrypts and decrypts the file dynamically in the driver layer, which has high efficiency. So file filter-driven transparent encryption is used in this paper.

3.1 File Filter Driven Transparent Encryption Technology

File Driver Encryption (IFS) technology is based on Windows File System Filter Driver (IFS) technology [5], which works in the kernel layer of Windows. Without affecting the upper and lower interfaces, it can intercept all file system requests, so that new functions can be added without modifying the upper software or the lower driver, as shown in Fig. 2 [4]. It is characterized by high encryption efficiency and security, but the technical threshold is high. It is necessary to understand the Windows system kernel in depth and difficult to develop. All tables and figures with text only should be boxed in; i.e., a box should be drawn around the table or figure either by hand with a ruler or with a draw facility on.

4 Design and Implementation of Transparent File Encryption System

4.1 Overall Design

This system is a transparent file encryption system based on Android platform. It mainly completes the encryption and decryption of specified files, and takes into account the user's good experience, so as to ensure the personal information security of Android users.

The frame design of the whole system is shown in Fig. 3.

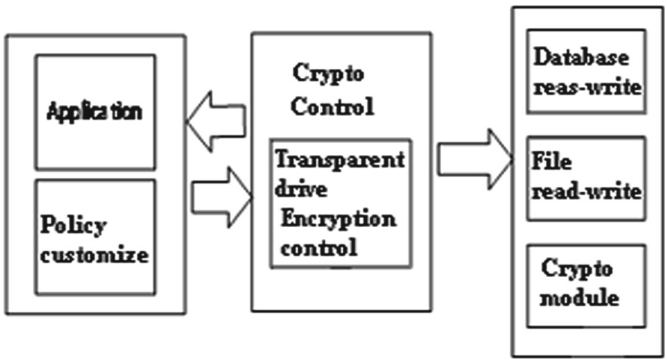

Fig. 3. System overall design framework

The system uses MVP (Model, View, Presenter) framework. Model (model) receives the control information from the controller, completes the operation of reading and writing files and encryption and decryption. View (user interface) mainly realizes the interaction with users and updates the user's encryption policy customization to relevant database items. Presenter is responsible for logical processing, customizing and updating the monitoring list according to the user's encryption strategy, monitoring and accepting the data read and write operations applied in the list, and passing the

information to Model. MVP is evolved from MVC framework [6]. It cuts off the connection between View and Model, makes View interact only with Presenter, increases readability and reusability, and reduces the cost of later testing and maintenance [7].

In the choice of encryption algorithm, we chose the currently popular encryption algorithm AES and the encryption algorithm SM4 independently developed by China. The reason for making these two choices is mainly considering the following reasons:

1) The reason why AES is selected as the encryption algorithm is because AES encryption algorithm is one of the most popular algorithms in symmetric encryption, and it replaced the original DES encryption algorithm. For now, AES is still a primary consideration in some encryption scenarios. At least for now, the AES encryption algorithm does not show a decline, nor does it show obvious problems in security;
2) The reason why I chose AES as the encryption algorithm for this design made in this article and developed a similar function using SM4, mainly considering that SM4 is an encryption algorithm independently developed by China. The emergence of encryption algorithms such as SM4 independently developed by China is an essential measure to ensure the security of encryption algorithms.

The system authenticates the user through screen lock when the mobile phone starts. When the system intercepts the user to read the file, it calls the function module of the kernel to decrypt the ciphertext, and then transmits the decrypted plaintext to the application layer for the user to read. When the system intercepts the user to write the file, it stores the plaintext encryption on the storage device to improve the user experience and security performance.

4.2 Design and Implementation of Encryption and Decryption Module

The performance of the encryption module affects the security of the transparent encrypted file system [8]. For the encryption algorithm, we have made the following two choices.

AES. AES is known as the advanced encryption standard. The AES algorithm requires a 128-bit or 16-byte length of plaintext, and the length of the secret key can be divided into 128-bit, 192-bit, or 256-bit (16, 24, or 32 bytes) [9]. The AES encryption process involves four operations: AddRoundKey, SubBytes, ShiftRows, and MixColumns. The decryption process is the corresponding inverse operation of the encryption [10]. Figure 4 shows the working flow chart of the encryption module.

SM4. Similar to DES and AES algorithms, the SM4 algorithm is a block cipher algorithm. The packet length is 128 bits, and the key length is also 128 bits. The encryption algorithm and the key expansion algorithm both use 32 rounds of non-linear iterative structure, and encrypt operations are performed in units of words (32 bits). Each iteration operation is a round of transformation function F. In SM4, the structure of the encryption algorithm and the decryption algorithm are the same, except that the

round key used is opposite, where the decryption round key is the reverse order of the encryption round key.

Regardless of whether AES encryption algorithm or SM4 encryption algorithm is used, the functions we want to achieve are consistent. Figure 4 shows the workflow diagram of the encryption module.

The mac address is unique to a terminal device, so the mac address is used as the encryption and decryption key. By reading the MAC address of the Android terminal, after a series of replacement transformations and other operations, it is transmitted to the encryption algorithm of the kernel module as the encryption key of the current device. Among them, accessing the MAC address of the Android terminal requires reading the address under / sys / class / net / wlan0. Therefore, each terminal has its own unique key. If the terminal is changed, the files of the local terminal will not be able to view. The function plays a vital role in protecting the privacy of mobile files.

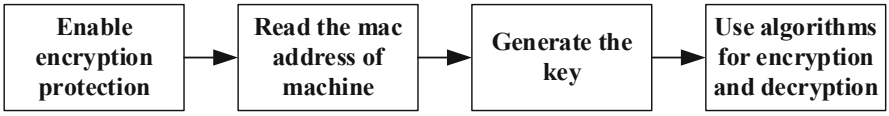

Fig. 4. Workflow of encryption module

4.3 Design and Implementation of the Whole System

The whole design module of the system is divided into application layer module and kernel module. The application layer module mainly completes the function of customizing encryption strategy and interacting with users; the kernel module completes the functions of monitoring, encryption and decryption, data reading and writing according to the setting of application module.

The overall design flow chart of the system is shown in Fig. 5. After the system starts to run, the user carries out the "policy customization" operation at the user level, enters the kernel layer after the policy formulation, and monitors the reading and writing operations of the files. In order to read a file, the first step is to determine whether the file is an open encrypted protected file. If it is, it decrypts and passes the data to the user; if it is to write a file, it is still necessary to determine whether the file is an open encrypted file, and if it is, it is encrypted and writes the data to the database or SD card. If the read-write operation file is not the file protected by the policy, then the normal read-write operation can be carried out.

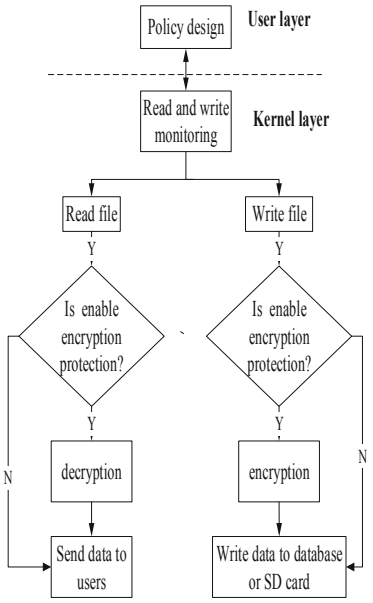

Fig. 5. The overall design flow of the system

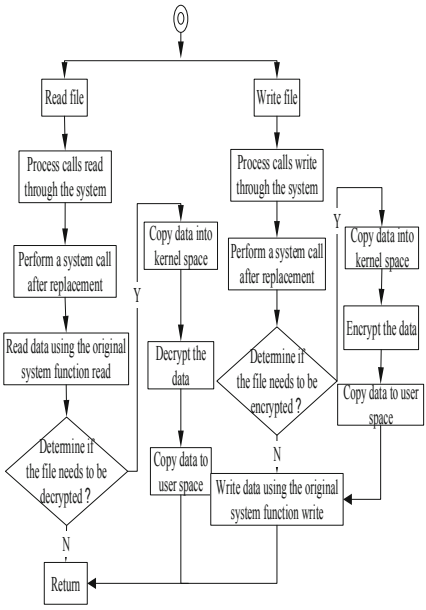

Fig. 6. The Design flow of kernel module

4.4 Design and Implementation of Kernel Module

System calls under Linux are implemented with soft interrupts. The interrupt program handles different system calls according to the system call number. Through the soft interrupt program, the program will be trapped in the kernel space for system call processing. In addition, Linux provides a program that can load kernel modules, namely LKM (Loadable Kernel Module), which is mainly used to dynamically extend the functions of the Linux kernel [11]. Figure 6 shows the workflow diagram of the kernel module.

When run the process of writing files, the encryption process, is executed, the interception system calls write. At this run point, the kernel module gets the file name and structure. Comparing with the file name and file structure in the encryption strategy formulated by the application layer, if the file name and file structure match, the data will be copied to the kernel space, and AES symmetric encryption operation is performed on the data through the pre-written encryption function. The key is obtained from the application layer and encrypted. Then the file is copied to the user space, and the data is written into the storage medium by calling the original system function write. If the current operation of the file is not specified in the encryption policy, the normal file write operation is run.

When the read file operation is performed, the interception system calls read. Get the file name and structure of the file currently operating in the kernel module.

Comparing with the file name and file structure in the encryption strategy formulated by the application layer, if the file name and file structure match, the file will be copied to the kernel space and decrypted. After the plaintext data is transferred to the kernel space, if the current operation file is not specified in the encryption policy, the file reading operation will be performed normally.

5 System Testing

The system is installed on API18 simulator and successfully implements the transparent encryption function of txt and doc file format on SD card. After the encryption is successful, the files can be viewed normally at the local terminal. Replacing the terminal and viewing it on the PC and another mobile phone is random code, thus completing the privacy protection of Android mobile phone files, as shown in Fig. 7, 8, 9, 10, 11 and 12.

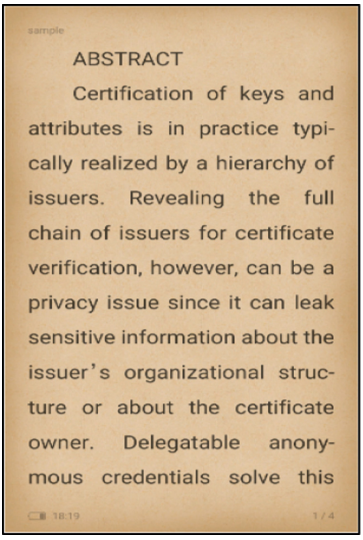

Fig. 7. The encrypted TXT on local terminal

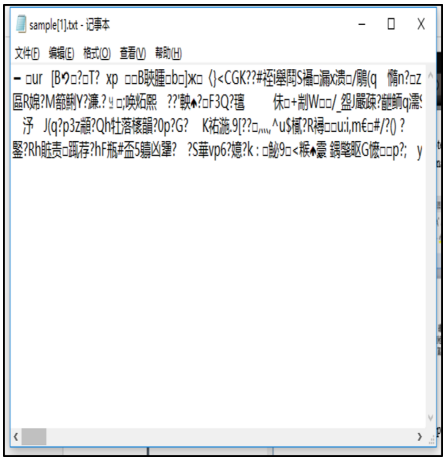

Fig. 8. The encrypted TXT file on PC

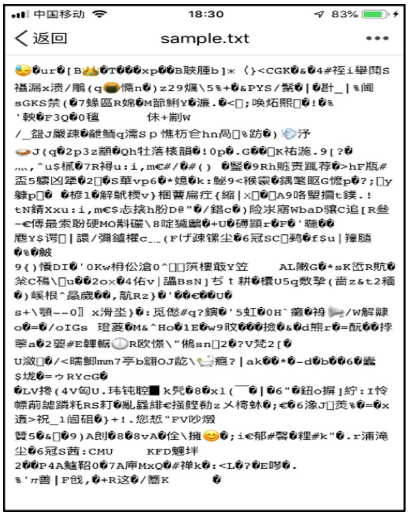

Fig. 9. The encrypted TXT file on another terminal

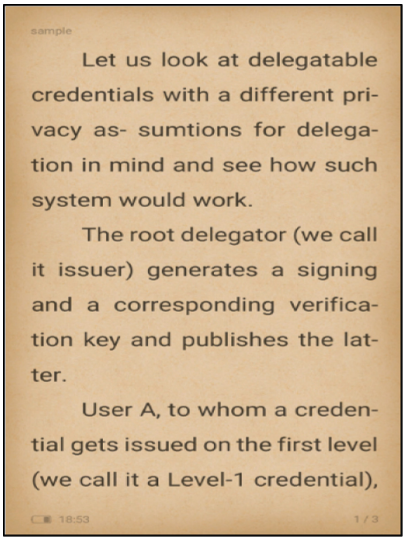

Fig. 10. The encrypted doc file on local terminal

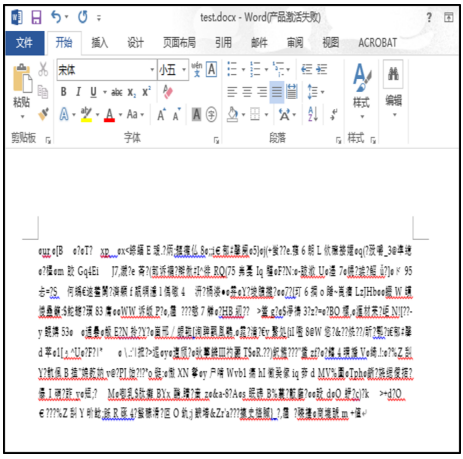

Fig. 11. The encrypted doc files on PC

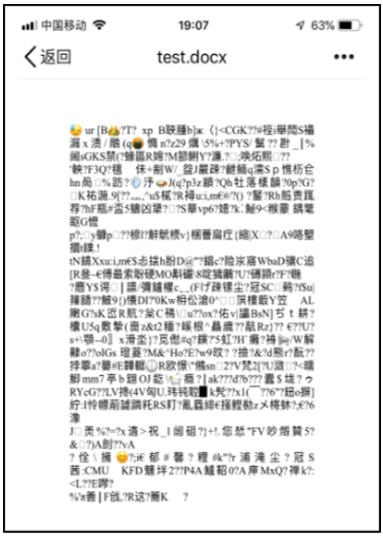

Fig. 12. The encrypted doc files on another terminal

6 Conclusion

According to the test results, when the encryption is completed, the files can be viewed normally on the local mobile phone after the authentication of screen lock, but not in other environments. The transparent encryption function of files under Android platform has been successfully implemented. The system uses file filtering to drive transparent encryption technology. By intercepting system calls and using keys converted from MAC address of host computer, the encryption and decryption algorithm is written into the kernel. In the process of encryption, the plaintext of files only appears in the kernel layer, which has the characteristics of security, stability and efficiency. However, the software user interface for this system can also be beautified, and the type of file data for encryption protection can also be increased, which will become the next research content.

References

1. Ma, L., Gu, L., Wang, J.: Research and development of mobile application for android platform. Int. J. Multimedia Ubiq. Eng. **9**(4), 187–198 (2014)
2. Enck, W., Ongtang, M., McDaniel, P.: Understanding android security. IEEE Secur. Priv. **7** (1), 50–57 (2009)
3. Yang, D., Peng, Y., Fang, Z.: The application of transparent decryption in trusted storage of electronic documents. Electron. Sci. Technol. **4**(04), 147–150 (2017)
4. Wang, Q., Zhou, Q., Liu, Y.: Research on file system transparent encryption techniques. Comput. Technol. Dev. **20**, 147–150 (2010)
5. Liu, W., Li, D.: A file protection scheme based on the transparent encryption technology. In: 2018 IEEE International Conference of Safety Produce Informatization (IICSPI), pp. 793–795 (2018)
6. Lin, Y.: Application of MVVM design pattern and MVP design pattern based on ZK. J. Chongqing Univ. Arts Sci. (Nat. Sci. Ed.) **72–74**, 78 (2010)
7. Zeng, L.: Application research of MVP for android. Comput. Eng. Softw. **2016**(06), 75–78 (2016)
8. Fu, C.: Design of transparent encryption system for documents based on windows kernel. J. Chongqing Univ. Educ. **28**, 171–173 (2015)
9. Nawaz, Y., Wang, L., Ammour, K.: Processing analysis of confidential modes of operation. In: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage (SpaCCS), pp. 98–110 (2018)
10. Qiu, P., Wang, D., Lv, Y., et al.: Voltjockey: breaching trustzone by software-controlled voltage manipulation over multi-core frequencies. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 195–209 (2019)
11. Salman, H., Uddin, M.N., Acheampong, S., et al.: Design and implementation of IoT based class attendance monitoring system using computer vision and embedded linux platform. In: Workshops of the International Conference on Advanced Information Networking and Applications (AINA), pp. 25–34 (2019)