Secure Storage Model Based on TrustZone

Shunrui Huang\textsuperscript{1*}, Chuanchang Liu\textsuperscript{1} and Zhiyuan Su\textsuperscript{2}

\textsuperscript{1}State Key laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
\textsuperscript{2}School of Automation, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding author e-mail: shunruihuang@gmail.com

Abstract. The explosive growth of the mobile Internet has brought great convenience to people's lifestyles, but it is followed by corresponding security issues. Especially on the Android system with open-source nature, once the system is rooted, it will not be able to protect effectively the user's data security. This paper proposes a secure storage model to provide better security for secure storage of private files based on TrustZone technology. We isolate the system's hardware and software environment into two areas-the Trusted Execution Environment (TEE) and the Rich Execution Environment (REE) by using TrustZone technology. We perform normal operations in the REE and switch to the TEE environment for encryption and decryption when it is necessary to operate the privacy file. We combine a traditional encryption and decryption algorithm to implement a secure encryption and decryption module to encrypt and decrypt private data. The core encryption and decryption operations are implemented in the TEE to achieve secure encryption and decryption.

1. Introduction

Due to the development of the mobile Internet, security issues are becoming more and more important in the field of mobile phones. Android has been criticized for security risks. The security risks of Android are mainly from malicious applications. Even if you download from the official app store, there will still be problems. Malicious apps attack other applications or steal user data through specific behaviors\cite{1}. Due to the seriousness of the security issue, Google has specified that after Android 7.0, vendors must use TEE to protect the user's biometric data (fingerprint, iris, etc.). To ensure the security of users' data, ARM has proposed TrustZone technology.

The most straightforward and simple way to ensure the security of stored data is to encrypt the stored data. Cryptography is an important means to achieve secure transmission and secure storage of data, ensuring that data is not tampered with and leaked. Data encryption is divided into two types: one is hardware encryption, and the other is software encryption. Since hardware encryption requires hardware cooperation, the implementation cost is relatively high and will not be discussed here. Software encryption is the use of a software encryption module to encrypt user information. However, since software encryption is performed inside the computer, it is easy for the attacker to crack the information using tracking and decompilation.

TrustZone technology provides two worlds - the normal world and the security world. The normal world refers to the common operating systems and their applications, the security world running security operating systems and security applications. Usually the security world runs a smaller kernel...
that provides security services. The switch between the two world is carried out by the control mode with the higher level - controller mode. Normal world applications can be switched to the safe world by the security monitor call (SMC) [2].

This paper has two main contributions. The first contribution is to propose a TrustZone-based security model that implements a secure isolation of privacy operations by building a TEE environment. The main purpose is to use the hardware security isolation environment built by TrustZone to achieve the isolation of the common execution environment, thus preventing other applications from tracking and eavesdropping the process during the operation. The second contribution is the design and implementation of the encryption module for private data. Combined with the traditional encryption and decryption scheme, the encryption and decryption of private data are realized in the trusted execution environment, which prevents the monitoring and tracking of malicious applications and prevents the leakage of private data.

2. Related Work

The key to ARM TrustZone is "isolation." That is, each physical processor core is divided into two areas: a non-secure core (also called a common area NS) and a security core (also called a security area); such that components of the normal area can only access common area resources and cannot access the security area. Non-secure cores can only access public resources, while the security core can access all resources [3]. ARM adds an additional secure bus to the AXI system bus, called the NS bit, and divides the cortex into two states: secure world, non-secure world, and adds a pattern called monitor, cortex determines whether the current instruction operation requires a secure operation or a non-secure operation based on the NS value, and determines whether the current instruction operation needs to be performed in conjunction with whether it belongs to the secure world state or the non-secure world state. The switch between the secure world and the non-secure world state of cortex is done by the monitor. Recently, due to the ATF (arm trusted firmware), the state switching operation of the cortex is completed in the ATF. When the cortex is in the secure world state, the cortex will execute the TEE (Trusted execute environment) OS part of the code. When the cortex is in the non-secure world state, the cortex will go back to execute the Linux kernel part of the code. The Linux kernel cannot access some of the resources of the TEE. Only the specific TA (Trust Application) and CA (Client Application) can access the specific resources of the TEE. TEE is a secure execution environment based on TrustZone technology. When cortex is in the secure world state, cortex executes the code of TEE OS. There is currently no unified TEE OS in the world, and each vendor and organization has its implementation, but the external interfaces of all solutions will follow the GP (GlobalPlatform) standard. So the use is more convenient for the second-tier manufacturers. The current vendors with their TEE solutions are Qualcomm's Qsee, Trustonic's TEE OS, OP-TEE OS, OpenTEE, and Hess Mstar.

Encryption techniques are generally divided into symmetric encryption and asymmetric encryption. Symmetric encryption means that the keys encrypted and decrypted are consistent. In the asymmetric encryption cipher algorithm, The keys for encryption and decryption are different. Each user's encryption key is public and also is known as a public key. All users' public keys will be recorded on a key book that all users can access. Correspondingly, each user's decryption key is saved by the user himself and strictly confidential, so the decryption key is also called a private key. The efficiency of symmetric encryption is generally high, but the disadvantage is that once the key is compromised, the information is completely exposed. Asymmetric encryption is relatively safer but relatively inefficient. The Advanced Encryption Standard (AES) is the most common symmetric encryption algorithm (the WeChat applet encryption transmission uses this encryption algorithm). The symmetric encryption algorithm uses the same key for encryption and decryption. AES, also known as Rijndael encryption in cryptography, is a replacement for the original DES and has been widely analyzed and widely used. AES has been widely used in practical scenarios and has high scientific significance and application value [4].
3. Design

3.1. Secure storage model based on TrustZone
The secure storage model is based on the security framework of the ARM Trust Zone. The encryption module and file system are added to the trusted execution environment (TEE). After the sensitive information is transmitted to TEE, it is encrypted by the encryption module and then passed through the file system. The encrypted ciphertext is stored on the secure storage chip; when the sensitive information is to be obtained, the file system reads the ciphertext on the storage chip, passes the ciphertext to the encryption module for decryption, and finally transmits the plaintext to the application. The secure storage model consists of two parts, one is based on TEE to build a TEE, and the other is the design of the privacy data encryption module. The structure of the overall secure storage is as shown in Fig. 1.

![Figure 1. The Structure of the secure storage.](image)

3.2. Construction of a trusted execution environment
In the GlobalPlatform standard, the hardware and software architecture of TEE has been stipulated accordingly. The ARM TrustZone security mechanism provides hardware support for TEE. The system architecture of TEE is as shown in Fig. 2 [5].

The Trusted Execution Environment software architecture enable Trusted Applications (TAs) to provide isolation and reliability to service providers through the intermediary client application (CA) to use.

The TEE Client API is used by the CA on the REE side, and the TEE Internal API is the standard API for TA to call Trusted OS resources. It is also used to facilitate TA developers to develop on different software and hardware platforms. The TEE Client API operates the TEE Driver through the ioctl system call. The TEE Driver is a bridge between the REE and the TEE. CA passes the data to the Linux kernel. The Linux kernel copies the user layer data to the kernel layer. The Linux kernel requests a shared memory from the daemon. The Linux kernel copies the data to the shared memory and converts the virtual address where the data resides into a physical address. The physical address is
placed in the parameter, and the interrupt is passed to the TEE. The TEE then converts the physical address to a virtual address, reads the data, and passes it to the TA.

3.3. Design of the cryptographic module

The design of the cryptographic module involves two parts, first part is the encryption of the original file, and the other is the management of all encrypted security files. The encryption of the original file needs to be combined with the traditional encryption algorithm, the most critical of which is the generation of the key. The generation of the key is related to the device, and the generated process and generated values cannot be obtained in the normal execution environment. The management of secure files requires the principle of high efficiency and security. Secure storage uses a binary tree to store encrypted files. When you first use secure storage to create sensitive data, you need to generate two files in a specific directory: the meta-info file and the file named by the number. The meta-info file stores all the directory information and node information of the file protected by the entire secure storage. When using an existing security file, you first need to read the relevant content in the meta-info file and then operate the security file according to the need. The hash value of the name is found in the meta-info file, and finally the file is opened, closed, written, read, renamed, cropped, etc. according to the number. Files named after numbers are specific user files saved by secure storage. This file holds the data after the user data is encrypted.

4. Implement

4.1. Build a trusted execution platform

The trusted execution environment is based on the open source project OP-TEE. OP-TEE is mainly composed of three parts: Opsee_client, Opsee_linuxdriver and Opsee_os. Opsee_client implements the ClientAPI specification defined by GP. Opsee_linuxdriver implements the driver module for accessing the TEE. Opsee_os is the core of OP-TEE and implements a trusted OS.
Building a trusted execution platform requires building a qemu+OP-TEE environment. OP-TEE is an open-source project that can be downloaded from GitHub to the local and compiled. The compilation of the entire project starts with the Makefile in the build directory or the corresponding board-level xxx.mk file. We need to write our TA and CA programs, first modify the code and makefile in the TA directory, put the TA code into the OP-TEE root directory, then write the CA part of the code, put it into the host directory of the TA program. Next, try to compile the TA and CA code. Next we need to integrate TA image and CA binary into rootfs and global makefile, then compile the added TA and CA into the entire project. If the compilation successes, you can try to call it to encrypt some security files.

The most important part of building a trusted execution environment is the writing and adding of trusted applications (TAs) and client applications (CAs). Add the designed TA program and CA program to OP-TEE. The main function of the TA and CA is to encapsulate the operation of the private data, including encryption and decryption operations.

4.2. Encryption module implementation
The encryption of the original file is encrypted by the AES algorithm. When using the AES algorithm for encryption or decryption, it is necessary to provide the key and initialization vector iv value used for encryption. Each TA will generate a random number as the iv value when using the secure storage function to save the data, and use the value of FEK as the AES key. The value of FEK is obtained by a series of HMAC operations. The generation of FEK values involves SSK and TSK, and the relationship between keys is as shown in Fig. 3.

**Figure 3.** Relationship between keys.

SSK: The value of SSK in each device is different. When OP-TEE is started, the chip ID and HUK are used to calculate the value of SSK through HMAC algorithm, and the value of SSK is saved in the general value in the structure variable tee_fs_ssk. The key members are formed to generate other keys. The HUK is written to the OTP/efuse during factory production, and the value of HUK cannot be read on the normal world side, and the chip ID is written to the chip after the chip is shipped.

TSK: TSK is used to generate the key used by FEK. The value of TSK is obtained by the UUID of the TA using SSK as the key. It is obtained by HMAC calculation. The value of TSK is obtained by the method similar to HMAC (SSK, UUID), and the tee_fs_fek_crypt function is called. It will then calculate the value of TSK. TSK will eventually be used to generate FEK, which will be used to encrypt data when it is saved using the secure storage feature.
FEK: FEK is the AES key used by secure storage to encrypt data. The key is generated randomly by using PRNG when generating the file. The generated FEK is encrypted by TSK and then saved to the head. enc_fek variable. Each time TA creates a secure file using secure storage, it generates a random number as FEK, that is, each security file in each TA has a FEK for encrypting the corresponding file data.

When using secure storage to read and write on saved security files, the meta-info file will be opened first, then the data area in the meta-info file will be read, and the meta-info file entry information of all the files saved in the entire secure storage will be obtained, using each meta-info file entry. The uuid and obj_id in the comparison with the uuid and obj_id of the security file to be operated to find the file number of the security file stored in the /data/tee directory.

After querying the file number of the security file, it is necessary to operate the security file, and also read the location of the data to be operated in security, calculate the block number, and then obtain the node id corresponding to the block to obtain the block iv. Then you can decrypt the block to obtain the plaintext data using the FEK value stored in the security file head and the iv value of obtained block, or to perform an encryption operation to write the ciphertext data into the block. Files saved with secure storage are saved in the same format, and the meta-info file is the same format as the actual file.

In secure storage, both meta-info files and security files use a binary tree to save file numbers or data blocks. If it is in the meta-info file, the meta-info file entry structure variable (ciphertext preservation) is saved in the data block, and the node information corresponding to the saved data block is saved in the meta-info file. The corresponding data stored in the meta-info file entry data block can be found in the meta-info file. There is also such a correspondence in the security file, except that the data block is no longer stored as meta-info file entry but actually needs to be saved. The first node is used as the root node of the meta-info file or security file.

5. Conclusion
The entire system implements a secure storage model of files based on TrustZone. The system builds a trusted execution environment based on OP-TEE+qemu, and realize encryption function combined encryption algorithm and tree-like security file management system. The encryption algorithm is based on the AES algorithm, which implements the generation and management of key keys. Unified management of all security files improves the efficiency of encryption and reduces management costs.

Acknowledgments
The work was supported by the National Natural Science Foundation of China (Grant No.U1536112) and Special Funds for Public Industry Research Projects of State Administration of Grain of China (Grant No.201513002).

References
[1] Y. Zhou and X. Jiang, “Dissecting Android malware: Characterization and evolution,” in Proceedings of the 2012 IEEE Symposium on Security and Privacy, 2012, pp. 95–109.
[2] Yalew S D, Maguire G Q, Haridi S, et al. T2Droid: A TrustZone-based dynamic analyser for Android applications[C]//Trustcom/BigDataSE/ICESS, 2017 IEEE. IEEE, 2017: 240-247.
[3] Jing L, Chunhua J, Xia Y. Design and implementation of security os based on trustzone[C]/Electronic Measurement & Instruments (ICEMI), 2013 IEEE 11th International Conference on. IEEE, 2013, 2: 1027-1032.
[4] Iyer S C, Sedamkar R R, Gupta S. A novel idea of video encryption using hybrid cryptographic techniques[C]//Inventive Computation Technologies (ICICT), International Conference on. IEEE, 2016, 3: 1-5.
[5] Platform G. Global Platform Device Technology TEE System Architecture, Version 1.1 [J]. Public Release, 2017.