Security Technology for Smartphones

● Yasuhiko Abe ● Hitoshi Ikeda ● Masafumi Emura

Service functions are implemented on smartphones by storing on them personal information, network-operator information, corporate information, and so on. Most smartphones use an open source operating system (OS), and anyone can obtain the OS source code; consequently, smartphone users are exposed to the threat of receiving fraudulent information from people with malicious intent. Aimed at countering this threat, safe smartphones—on which information is protected by applying multiple technologies such as Secure Boot, Linux Security Module, and TrustZone™—are marketed. Secure Boot is a technology for detecting whether a boot program of a smartphone has been rewritten and protects network-operator information by making it impossible to start up programs modified by a third party. Linux Security Module prevents leaks of personal information by controlling access to information by any program installed after factory shipment and by stopping unauthorized access to that information. TrustZone ensures banking services are safe by separating program operating environments with hardware and by preventing unauthorized access to information by programs running on open source OSs. In this report, these technologies, namely, Secure Boot, Linux Security Module, and TrustZone, are described in detail.

1. Introduction

What comes to mind on considering the term "smartphone security technology"? People might think of device locking by means of passwords, biometric-authentication technology such as fingerprint identification, or perhaps anti-virus technology. To consider security technology thoroughly, it is necessary to think about how smartphones are used. For example, they are used as alarms in place of alarm clocks, for checking e-mail, for credit settlement after weekend trips, for paying train fares via mobile-phone services like "Mobile Suica," for reading electronic editions of newspapers via mobile applications when commuting, for playing games on the way to work, for efficiently moving from place to place while out and about, and for buying boxed lunches by "mobile wallet phone." Their ability to connect to the company e-mail server from any location enables a smartphone user to send reports to his or her manager and confirm instructions. After returning home, the user can access the Internet using a smartphone and purchase anything he or she wants. These are probably the ways most people use smartphones these days.

These functions are made possible by utilizing personal information stored on the smartphone (such as data in address books, application-purchase information, location information, and credit information), telecommunications carrier information (phone numbers, etc.), and corporate information (in-house server access information). But what would happen if a stranger used such information? The answer is that financial loss and/or damage to public trust would likely occur. Even if they lose their smartphone, most users would not worry because they know the phone is in lock mode. Many makers of smartphones, including Fujitsu, have adopted open source OSs—for which the technical data is publically available and there is little difference from the OSs of feature phones. It is thus possible to, for example, rewrite programs on smartphones and install lock-release applications. As a result, the information stored on a smartphone could be used by someone with malicious intent. What is more, it is

possible that such information could simply flow from the smartphone without the user's awareness.

In this report, the security technologies implemented in Fujitsu's smartphones—for preventing outflow of important information and its unauthorized use—are described.

2. Overview of security technology

1) Secure Boot

A person with malicious intent can modify a program in a fraudulently acquired smartphone if it uses an open source OS by first obtaining a copy of that freely available OS. Then, after modifying the program, the person can install it on the smartphone and use it to extract various kinds of information from the smartphone. By applying Secure Boot (explained in detail later) to a smartphone to counter this threat from illicit programs, Fujitsu can guarantee the integrity of the phone's OS and firmware. At smartphone start-up, this technology verifies the integrity of the programs and stops start-up by modified programs and secures the information in the device.

2) Linux Security Module

Smartphones can use applications purchased and installed by the user. In contrast to the case of feature phones, applications can be created on a smartphone and registered on an application market without being properly screened. Another user can then buy that application and install it on his or her smartphone. Fujitsu protects information by using Linux Security Module (LSM) technology to restrict operation of the application programming interfaces (APIs) used by applications so that data cannot be fraudulently acquired by purchased applications.

3) TrustZone™ note)

TrustZone technology (which isolates the program operating environment from the open source OS environment by using hardware modules) is also implemented in order to protect information from illicit programs.

Smartphones made by Fujitsu are fitted with various technologies other than those mentioned above (such as secure storage and manipulation detection) for protecting important information, but the three security technologies summarized above are described in more detail hereafter.

3. Secure Boot

The software installed in a smartphone should not be modified after shipment because doing so could nullify some of the protection measures implemented during manufacture. Even if the phone is protected by anti-virus software against malware and by LSM against illicit access, tampering with its software can weaken the security functions of these protection technologies.

Secure Boot uses digital signature technology to detect software tampering, and it prevents the execution of software that has been tampered with. A digital signature—created through a combination of a public-key cryptosystem and hash technology—is used for confirming that data exchanged between the data creator and the data user has not been illicitly modified.

In the Secure Boot process, a digital signature of the software (i.e., a "program image") is created and written into the smartphone. A function verifies the signature of the program executed on boot-up (i.e., the boot-loader). That is, the boot-loader reads the bootstrap program image and verifies the digital signature, thereby ensuring the integrity of the program.

The Android OS has a Linux kernel, and smartphones running Android are generally designed in such a manner that multiple boot-loaders are executed sequentially after the power is switched on, before the Android system is started up (**Figure 1**). The image of the primary boot-loader is stored as a non-rewritable program in read-only memory (ROM). It verifies the image of the secondary boot-loader by verifying the digital signature that has been created and stored in advance. The secondary boot-loader verifies the image of the Linux kernel in the same way.

3.1 Procedure for creating digital signature

The creator of the digital signature generates a private key and a public key in advance. A hash value, called a "message digest," is calculated from the target program image by using a hash function. The creator then encrypts the message digest by using the private key. The encrypted message digest is used as the digital signature. Subsequently, the program image and digital signature are written into the internal memory of the smartphone (**Figure 2**).

In addition, the public key must be stored in

note) TrustZone is a trademark of ARM Ltd.

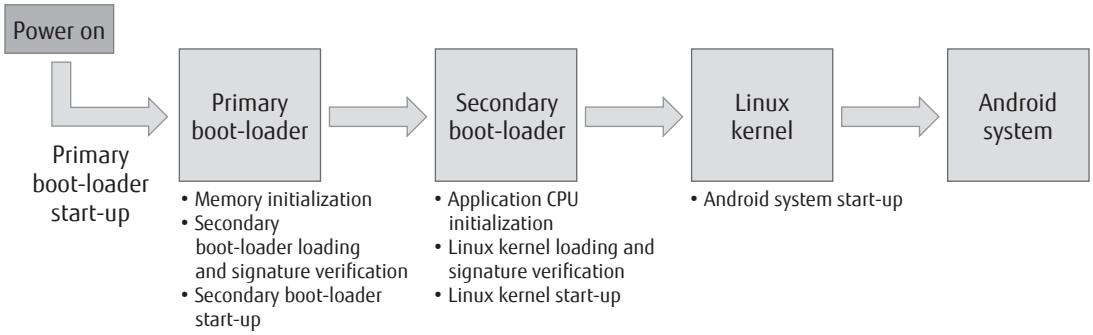


Figure 1
Sequential execution of multiple boot-loaders.

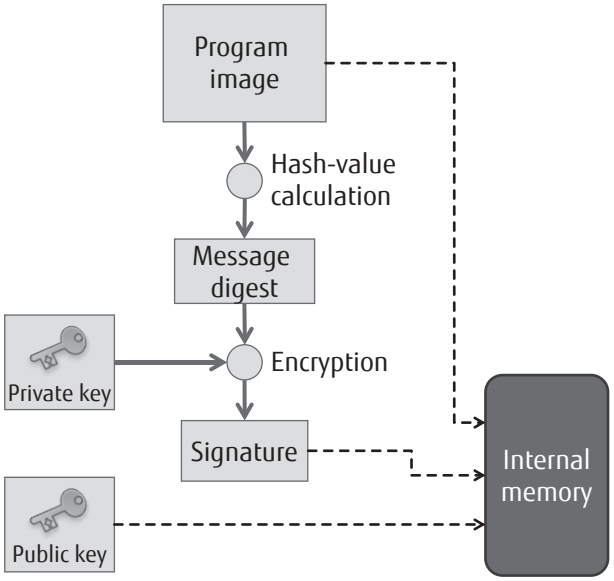


Figure 2
Signature creation process.

advance into a memory region that cannot be rewritten. This is because the reliability of the digital signature verification would be compromised if the public key itself were rewritten. An example of the memory arrangement in a smartphone is shown in **Figure 3**.

3.2 Procedure for verifying digital signature

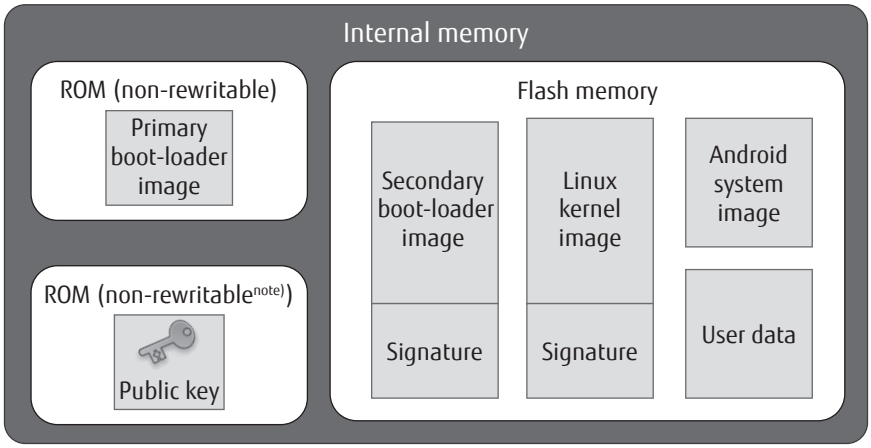
When the smartphone is switched on, the primary boot-loader (stored in ROM) is executed. As shown in **Figure 4**, it reads the image of the secondary boot-loader (stored in flash memory) and acquires the digital signature and public key. It calculates the hash value of the secondary boot-loader image and compares that value with the message digest in the digital signature (which is decrypted using the public key). If the values match, the image read from the flash memory is judged to not have been rewritten, and control is switched to the secondary boot-loader. If they do not match, the image is judged to have been rewritten, and the start-up process is terminated. The Linux kernel is verified in the same manner.

3.3 Operation rules for creating digital signature

If the private key is leaked, digital signature data could be created by a person with malicious intent. To reduce the risk of leakage, an operation rule for creating signatures should be strictly enforced. For example, the signature creators should be required to work in a secure location, one that is physically separated from the developers, when they create signatures. At Fujitsu, we have applied such rules.

4. Linux Security Module (LSM)

The LSM framework extends the security functions for kernels installed from Linux kernel version 2.6. It can summon from a developer-registered security module the call-back functions corresponding to the various system calls to the kernel. The security module is fitted with a security-check function in accordance with the security policy of the developer, and a return value can be set that enables the user to control access to the execution of system calls. LSM operation is shown schematically in **Figure 5**. The security module installed in Fujitsu smartphones prevents the LSM from falsifying the system even if root access is acquired. In the following subsections, several types of security modules are described.



note) Public key can also be stored in ROM (which can be written to only once).

Figure 3
Example memory arrangement in smartphone.

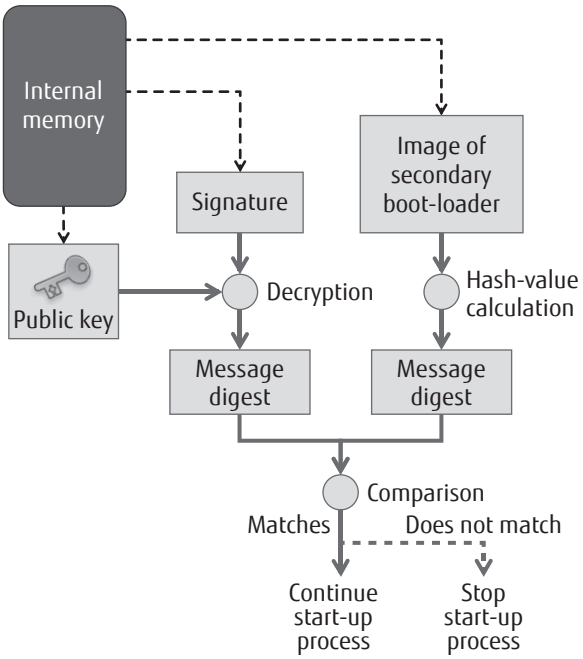


Figure 4
Signature verification process.

4.1 Inhibiting tampering with main disk

Controlling system calls (such as "write" and "pwrite") that execute writing targeting device files on the main disk makes it possible to prohibit access using system calls such as "io_submit," "fallocate," "link," and "rename." However, it can also make system updating difficult. For example the main-disk-falsification-prevention function prevents the use of "firmware over the air" (FOTA)—which updates system image files. Accordingly, if specified conditions are satisfied, write access to the main disk is permitted.

4.2 Other controls

1) Restriction of additions to kernel module

Installable kernel modules are controlled, and the addition of anything other than the previously determined kernel module is prohibited. The security module is called by the hooking process of an "init_module()" system call (which can be called by an "insmod" command). The security module registers the names of kernel modules and information for use in refusing registration of manipulated kernel modules. As a result, unregistered kernel modules and manipulated kernel modules cannot be added.

2) Restriction of holder of device files and authority

The holder and authority with respect to each device file under location "/dev" are determined in advance, and processes that change the holder or authority in regard to disk files are prohibited.

3) Restriction of mounting and fixing mount points

"Mount," "remount," and "umount" commands (for writing and rewriting) are prohibited, and tampering and access under specified mount points are inhibited. The mount points are fixed on a specified path.

4) Prohibition of changes to root file systems

Prohibiting changes to root file systems using certain system calls ("chroot," "pivot," and "root") makes it possible to deter changes to userland software.

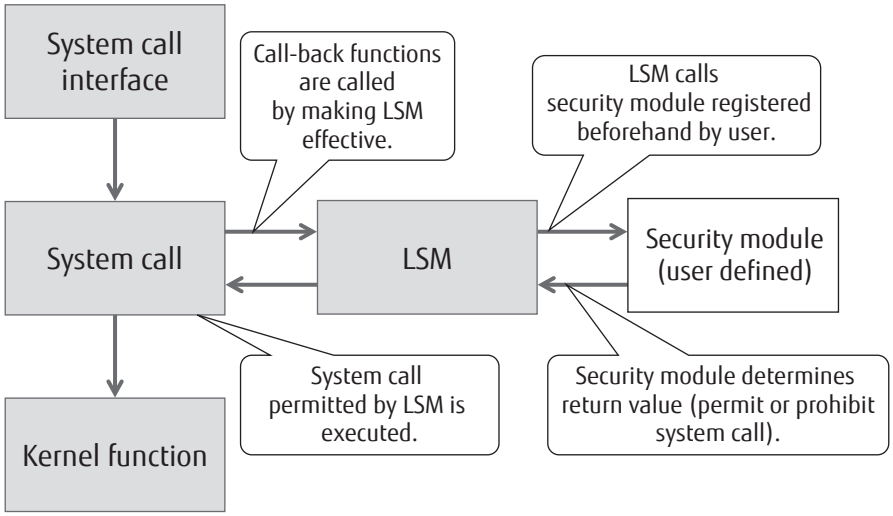


Figure 5
Configuration of LSM function.

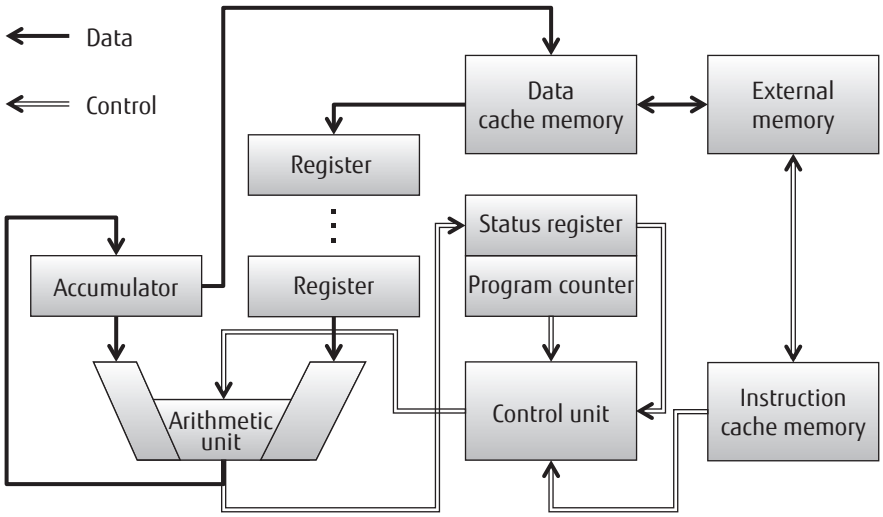


Figure 6
Configuration of Smartphone CPU.

5) Control of copying

File copying is prevented by prohibiting read and write accesses from or to files and directories at the hook point of each system call. When access from all processes is prohibited, normal use of files and directories is impossible; accordingly, access by specified processes only is allowed.

5. TrustZone

The CPU used in a smartphone is configured as a "serial processing unit," as illustrated in **Figure 6**. The instruction code is imported from cache memory in accordance with a program counter (expressing execution location), the code is fed into the control unit, data complying with the instructions are set in the calculation register (accumulator) and processed by the arithmetic unit, and the calculation result is output to the accumulator and sent to data cache memory. The appropriate bits in the status register (expressing the status of the CPU) are set in accordance with the calculation result (zero, overflow, minus, etc.), and the control unit refers to those bits and determines the next

command to execute. Complex processing is achieved by repeating this process.

When an interruption occurs in a system requiring real-time characteristics, the serial processing performed up to that time is stopped, and predetermined processing that corresponds to the cause of the interruption is instantaneously executed. The instantaneous processing at that time involves sidetracking the register group (shown in Figure 6), retaining the status before the interruption, and jumping to the head of the interruption. When the interruption process finishes, the sidetracked register group is restored, and the processing that was being executed on occurrence of the interruption is restarted.

TrustZone generates actions corresponding to these interruption actions in a software manner and creates new operational space that is distinct from the interruption. TrustZone (an implemented hardware technology from the v6 architecture of ARM, Ltd.) controls the process accesses by TrustZone and non-TrustZone software, separates resources, and preserves them in response to memory-regulating hardware-resource access (such as by a memory management unit [MMU]). When this technology is applied, software on open source OSs is placed in a non-TrustZone area, and reliable processes for accessing important data are placed in a TrustZone area. As a result, access to important data and hardware resources by open source OS processes is prohibited, and the information is protected. By making the decrypt-processing hardware accelerator usable from only the TrustZone area, the decrypt processing can be controlled. By ensuring that abusive applications cannot perform decryption, digital rights management (DRM) can be supported.

6. Conclusion

Smartphones can hold both personal and company information. Furthermore, unlike PCs, they can hold telecommunications carrier information. While open source OSs make it possible to continually improve product quality, their software code is open to the public—including persons with malicious intent. With the spread of smartphones using open source OSs, risks such as illicit access to the data they hold are growing. Technology for protecting information is thus required, and various such technologies have been developed. As a developer of these technologies, Fujitsu is providing smartphones that not only protect the user's information but that are also protected against illicit programs. Three key technologies (described in this report) that ensure information security are Secure Boot (which prevents the addition of illicit programs), LSM (which prevents information acquisition by illicit applications), and TrustZone (which separates execution space environments). Technologies used by persons with malicious intent for fraudulently acquiring information are, however, evolving; accordingly, security technologies have to evolve to keep up with them. Moreover, new open source OS vulnerabilities are being regularly discovered, and it is essential to eliminate those vulnerabilities in order to prevent their malicious exploitation. Fujitsu will thus continue developing security technologies, implementing them in our products, and developing secure smartphones in such a manner that we will contribute to realizing a society in which people can live in peace.

Yasuhiko Abe
Fujitsu Ltd.
Mr. Abe is engaged in the development of smartphones.

Masafumi Emura
Fujitsu Ltd.
Mr. Emura is engaged in the development of smartphones.

Hitoshi Ikeda
Fujitsu Ltd.
Mr. Ikeda is engaged in the development of smartphones.