Hindawi
Security and Communication Networks
Volume 2022, Article ID 6731277, 12 pages
https://doi.org/10.1155/2022/6731277

Research Article

A Novel Trusted Software Base for Commercial Android Devices Using Secure TF Card

Yuting Zhou,[1] **Bo Zhao**,[1] **and Yang An**[2]

[1]*School of Cyber Science and Engineering, Wuhan University, Wuhan, China*
[2]*School of Computer Science, Wuhan University, Wuhan, China*

Correspondence should be addressed to Bo Zhao; zhaobo409@126.com

Received 15 September 2021; Revised 17 December 2021; Accepted 26 January 2022; Published 16 February 2022

Academic Editor: Ding Wang

With the help of edge computing, the Internet of Things (IoT) provides users with efficient data transmission and processing capabilities. As a main control device of the IoT and the communication portal of edge computing, user terminals represented by Android devices have potential security risks in IoT. Trusted computing is a universal method to construct trusted environment for computing platforms. However, due to the strict space, cost, and power limitations, commercial Android devices would not be applicable to implement a dedicated onboard chip or the software Trusted Platform Module (TPM) by modifying its firmware. We have designed a practical Trusted Software Base (TSB) for mobile devices to enhance their security. By using the secure TF card as the hardware to provide secure storage and cryptographic capabilities, we implement the trusted boot and trust extension for applications on a commercial device to verify the feasibility of the TSB to ensure a trusted environment for users. Our implementation does not require any modification to the firmware or any additional hardware other than the secure TF card. Experimental evaluation shows that our method can provide trusted computing capability for commercial Android devices with low performance overheads.

1. Introduction

The Internet of Things (IoT) uses edge computing close to the geographic location of users or data sources to provide more reliable and fast services. Mobile devices serve as the main IoT control center and the communication portal for mobile edge computing. Especially, existing works reveal that Android devices face many practical risks [1–3]. For example, in the scene of the Internet of Vehicles, if the vehicle control application of the Android device is tampered with, the vehicle will become unavailable and the privacy information may also be leaked.

Android's dominance in the global smartphone market has resulted in a significant increase in the number of malware, which pose a huge threat to the Android devices' security. Researches mainly focused on the detection and analysis of malicious applications [4–7]. These approaches contribute to the protection of applications and data on the Android platform, but the security during trusted boot is not guaranteed. Due to the lack of trusted hardware, the security software on the device may also be maliciously tampered with.

Trusted computing technology establishes a specific integrity measurement mechanism in computing system [8] and builds a chain of trust from root of trust to the whole software and hardware [9]. It enables the computing platform to have the ability to distinguish whether the program code is trusted when it is running, thereby establishing effective prevention methods and measures for untrusted program codes [10]. The most extensively implemented form of trusted computing on end-consumer devices is the Trusted Platform Module (TPM), a dedicated hardware chip that offers facilities for cryptographic coprocessing, protected credentials, secure storage, or even the attestation of its host platform's state. The special benefits provided by TPM have always relied on the implementation of TPM as a "local trusted third party" on the device. The use of TPM to implement the PC trusted computing platform has been very mature, and in the latest Windows 11, Microsoft requires a TPM module in the device [11].

However, limited by space, cost, and power consumption, the implementation of TPM on mobile devices is to build a hardware-based trusted execution environment (TEE) [12–14], like ARM TrustZone where the TPM is implemented as protected software application inside the TEE. But there are still some attacks against trusted software in TEE [15–19].

On commercial mobile devices, the firmware is customized by the vendor and hard to be modified. The solutions mentioned above have only been verified in the development board or simulation environment. Some need to add onboard hardware or modify the firmware. Therefore, for users who cannot trust both the vendor and applications in the market, a security protection solution that can be conveniently used on commercial Android devices is critical. So, commercial device users need a convenient security solution without changing the structure of the device.

In order to solve these problems from a new perspective, in this paper, we offer an alternate implementation of the TPM based on the secure TF card combined with a well-designed Trusted Software Base (TSB) [20] to achieve the same capabilities as TPM without any changes to the hardware structure and firmware of commercial Android devices. A secure TF card as secure hardware should have encryption and secure storage capabilities. Although the SoC in the device already has encryption instructions, the encryption engine and secure storage space of the dedicated hardware can ensure the security of the calculation process and the keys. We implement our TSB on Xiaomi Mi Pad 4 and use the JW1292T8I-SDK TF card produced by Chengdu JAVEE Microelectronics. Applying a star style trust measurement model, we extend the trust from the TF card to the device when booting and ensure the trust of the application when launching or installing. To summarize, our main contributions are as follows:

(i) Under the condition that changes can hardly be made to the firmware and hardware of commercial Android devices, we use secure TF card to achieve a trusted boot. Using a secure TF card as physically separate hardware to perform cryptographic calculations and securely store critical information will be more secure than the implementations without dedicated hardware.

(ii) Besides boot process, our TSB guarantees the safety of applications for users. Since Android devices do not reboot frequently and are mainly threatened by malicious software, we design TSB so that it can monitor and control the launch and installation of applications in real time to prevent malware.

(iii) By using the modified star style trust measurement model, TSB will take over the computing task after the Primary Trust Base confirms that it is trusted. This reduces the burden of the TF card and provides an accelerated measurement speed.

(iv) We conducted experiments using Android devices that are available on the market rather than a reference board. The evaluation shows that the device can complete a trusted boot in about two minutes. Also, the TSB has caused a minimum delay of less than 0.3 s for the application launch or installation and has little drag on the performance.

The rest of this article is organized as follows. Section 2 shows the background and related works. Section 3 presents an overview of the TSB design. Section 4 introduces our Android TSB implementation based on a secure TF card. Section 5 presents the experimental evaluation. Section 6 concludes this paper.

2. Background and Related Works

2.1. Background. Trusted computing technology guarantees the security of information systems by building a chain of trust. The way to obtaining trust is to examine two interacting entities. When there is no direct interaction between two entities, it is necessary to build a chain of trust through a third entity to extend trust. TCG had defined three roots of trust: root of trust for measurement (RTM), root of trust for storage (RTS), and root of trust for report (RTR) [21]. The RTM is responsible for performing reliable integrity measurements, the RTS maintains integrity digests values, and the RTR can reliably report information held by the RTS.

As seen in Figure 1, a serial chain connects the BIOS Boot Block, BIOS, OS Loader, and OS. BIOS Boot Block is the RTM. The hash value is generated from the concatenation of the current hash value and a new value. Then, the hash value is saved in the platform configuration register (PCR) as the new integrity measurement value. Equation (1) describes this process, where the symbol ‖ stands for concatenation and the NewValue stands for the content of entity i.

$$\mathrm{NewPCR}_i = \mathrm{HASH}\left(\mathrm{OldPCR}_i \| \mathrm{NewValue}_i\right). \tag{1}$$

However, there will be losses in the process of trust transmission [22]. The longer the path of trust transmission is, the greater the loss may be. In addition, the traditional chain of trust measurement model is inflexible to add and delete components. Furthermore, embedded systems have strict requirements on functions, reliability, cost, size, and power consumption. Therefore, as shown in Figure 2, the star style trust measurement model is proposed [23], which adopts a star style chain of trust structure on embedded devices.

According to the star trust structure theory, the trust extension path of the star style chain of trust is shorter and more robust compared to the TCG chain of trust, which makes it easier to add or delete the component software version upgrades in the star style chain. However, the disadvantage is that the root node needs to measure the integrity of each node in the boot chain and make a judgment, so the computing pressure on the root node is heavy. Considering that hardware and firmware of commercial devices can hardly be changed, we modified the star trust structure to better adapt to TF cards and commercial Android devices. In our design, the TSB is the central node of this structure, and the details are shown in the next section.

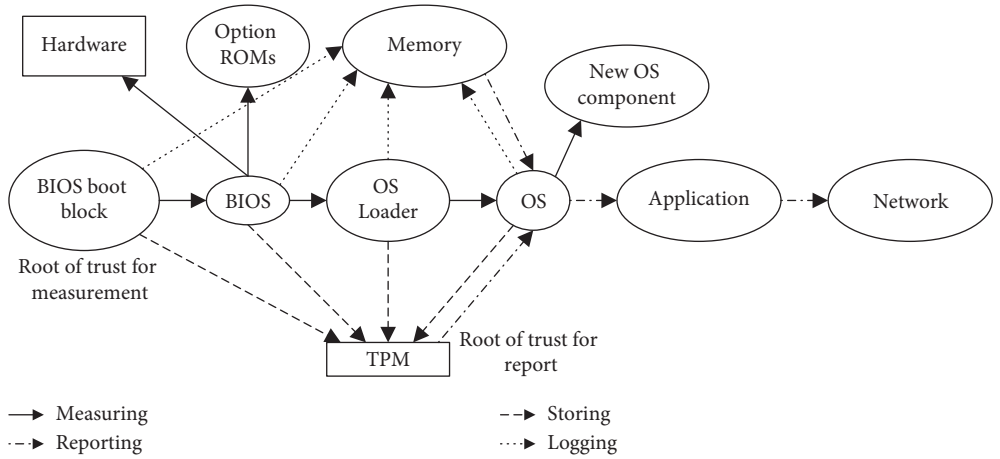

Figure 1: TCG chain of trust structure.

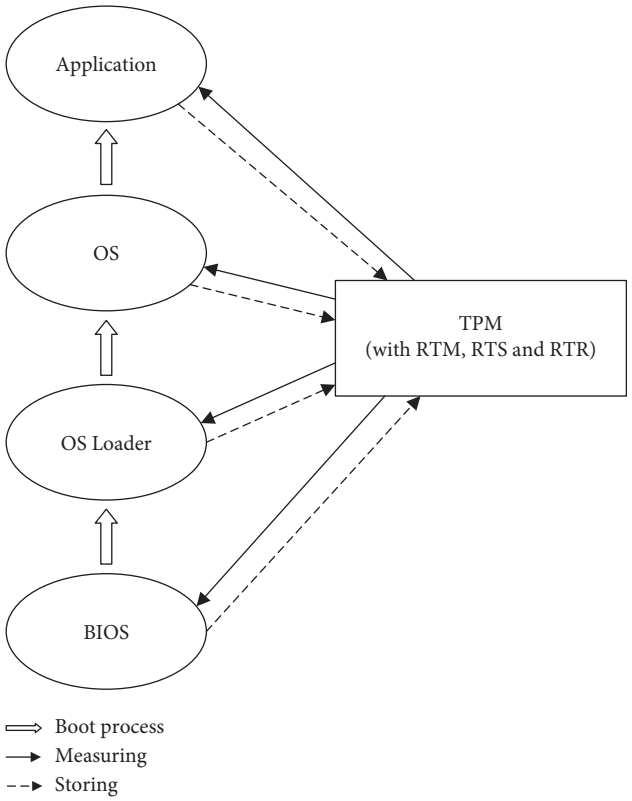

Figure 2: Star style chain of trust structure.

2.2. Related Works. Researchers have actively investigated how to bring trusted computing to mobile platforms. As mentioned earlier, Zhao et al. proposed an embedded system-TPM and a star style trust measurement model [23] to address the specific problems of the mobile domain. They mentioned that this model is better suited to the working environment of embedded devices and applied bus arbitration technique to manage the TPM and the embedded CPU to improve the security and efficiency. This work inspired our designs on mobile devices with limited resources, which draws on star style trust measurement model. In order to apply trusted computing technology in the mobile field, the Trusted Computing Group (TCG) developed the Mobile Trusted Module (MTM) specifications [24], which has an impact on the formulation of the TPM2.0 specification [25]. The TPM 2.0 mobile reference architecture [26] proposes solutions such as virtualization and hardware-based isolation to implement TPM on mobile devices. Ekberg et al. presented onboard credentials [12] to safely open access to TEE functionality on mobile devices. McGillion et al. described Open-TEE [13], a virtual, hardware-independent TEE implemented in software on mobile devices. Raj et al. presented fTPM [14], an end-to-end implementation of a software-only TPM using ARM TrustZone, which provides security guarantees similar, although not identical, to a discrete TPM chip.

However, due to the lack of dedicated hardware, attacks against trusted software in TEE can be implemented. Lipp et al. demonstrated cross-core caching attacks on non-rooted devices to steal secrets in TrustZone [15]. Machiry et al. discovered the boomerang [16] vulnerability in the TEE in the mobile platform, which allows untrusted applications to leverage the TEE's privileged position to perform the operations on its behalf. Ning and Zhang [17] discovered a security flaw in the arm debugging architecture, which can lead to arbitrary access to TrustZone.

Chakraborty et al. implemented a TPM called simTPM [27] without additional hardware by implementing TPM2.0 functionality on a subscriber identity module (SIM) card. Their work made it possible to use dedicated hardware on mobile devices to implement TPM-related functions.

The TSS specification [28] defines the way software at different layers uses TPM functionality, but does not establish a unified software architecture to provide trust support for devices.

Shi defined the overall trust support for system software by proposing the idea of Trusted Software Base (TSB) [20], which inspired the work of this paper to some extent.

3. Overview

3.1. Threat Model. In our threat model, the attacker needs to control the installation of malware on the victim's device in order to be able to carry out the attack. This is a very

common threat model in the field of mobile security, as in previous work [29–31]. On Android devices, it is easy for attackers to entice victims to download malicious applications that they have prepared in advance.

As the control and proxy device of the IoT devices, mobile phones will cause the IoT devices to deviate from the preplanned work plan once they are controlled or unusable. Attackers entice victims to install and use malicious software in order to damage the Android device itself and cause information leakage of the device or IoT device. Malware can also modify the IoT management application itself or its configuration on the Android device to attack the IoT system. Attackers can also destroy the availability of IoT devices through mobile phones infected with malware, such as denial of service attacks, energy exhaustion attacks, and even directly shut down the IoT system.

3.2. Trusted Software Base and TF Card. TSB is the fundamental link that connects trusted hardware to user space applications [20]. TSB protects user space applications and manages TPM and undertakes the extension of the TPM trust chain. With the support of trusted hardware, TSB implements security capabilities in the host system through measurement and interception.

In Figure 3, TSB is composed of a defense mechanism and an underpinning mechanism. The defense mechanism includes Measurement Mechanism, Control Mechanism, Decision Mechanism, Baseline Repository, and Primary Trust Base. The TF card provides the foundation for effective TSB operation by providing a tamper-resistant hardware root of trust. Our TSB provides the following two services to ensure the trusted environment of the device. First, TSB measures the boot chain of the system during the boot process to ensure a trusted environment after booting. Second, after the device boots up, TSB measures the launch and installation of applications in the device in real time to prevent untrusted applications from running. We will not leave any confidential information in the storage. But we do not protect the key content in the memory in this process.

The main goal of the TSB is to examine whether the host system is trusted. TSB itself must be trusted and tamper-resistant. Software measurement alone is not enough to meet the requirements of tamper-resistant [32]. We avoid using the method of changing original hardware on commercial devices. Since most Android phones that support dual sim cards are compatible with TF cards, the secure TF card is our final choice.

Our purpose of using a secure TF card is to use dedicated hardware to compute and store in the case of unchangeable hardware and firmware of commercial devices. Like TCG's TPM specification, the secure TF card should provide secure storage, public key and symmetric key encryption engine, hash engine, and a random number generator, as shown in Figure 3. These cryptographic engines and the random number generator exist in the main control chip of the TF card in the form of hardware and provide external calling

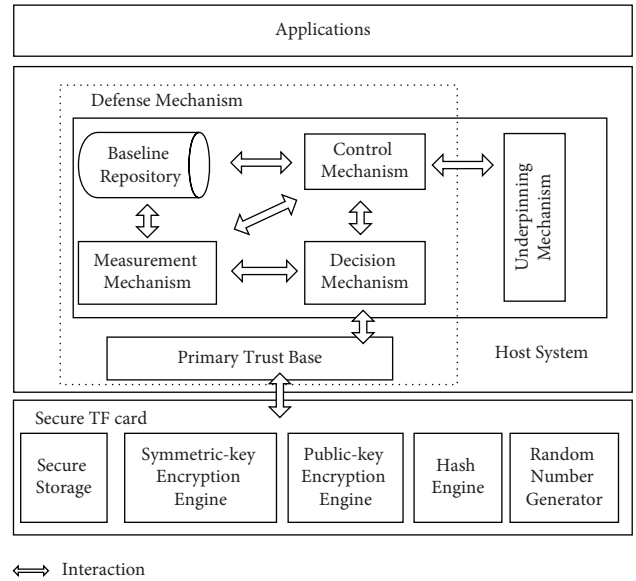

Figure 3: TSB structure.

interfaces. The information of the TF card used in our experiment will be described in more detail in Section 5.1.

The key used for data encryption is generated and managed by the TF card. Secure storage provides space for the password to the Baseline Repository and the whitelists formed by correctly measured values of all items to be measured and the PCR value of the boot chain. The content that needs to be safely stored in the TF card is encrypted by a symmetric key. These symmetric keys are generated and protected by the TF card. The TF card uses its built-in public key to encrypt and save these symmetric keys. Since the TF card is not mounted normally, ordinary users cannot operate the TF card without the system permissions that TSB has. In addition, the interface of the TF card is not exposed, and the authentication method is secretly shared by the TF card and the TSB. Therefore, the TSB and the TF card are strongly bound, and the outside world cannot access the TF card through this unique method.

Due to the limitations of commercial Android devices, our implementation of TSB is different from traditional methods. Nevertheless, it still contains the essential mechanisms that TSB should have. We will explain in detail the function of each part of our design and the specific process of trust extension in the next section.

4. Methodology

4.1. Trusted Software Base Structure. We will describe each part of TSB and how they work together.

4.1.1. Primary Trust Base. Primary Trust Base is the essential component in TSB and the most miniature set of scalable software with basic measurement capabilities. With the support of the underlying hardware platform (the secure TF card in this paper), it extends trust to other parts of TSB. It

completes the most fundamental measurement work when other mechanisms in TSB have not been confirmed as trusted yet. It implements the measurement operation to TSB by calling the interface of the TF card. This process passes trust to all remaining parts of TSB. In order to reduce the calculation pressure of the TF card and increase the calculation speed, we regard TSB instead of the TF card as the central node of the star style chain of trust structure. Then, TSB will be responsible for measuring all other entities in the star style chain.

4.1.2. Control Mechanism. Then, the Control Mechanism decides whether to allow, block, and audit the behavior of the measured object in accordance with the control policies and the result of the Decision Mechanism.

The control policies include (1) blocking the installation of apps that are not in the whitelist; (2) preventing the tampered application from running; and (3) prohibiting booting of devices with modified critical files.

4.1.3. Measurement Mechanism. Because of requirements for power consumption and real-time performance in Android platform, a lightweight Measurement Mechanism is required. The Measurement Mechanism is based on the integrity measurement model proposed by TCG. The measurement process refers to calculating the hash value of the object through the measurement algorithm and then comparing it with the expected hash value in the TF card. If the two values are the same, we can say that the measured object is trusted. The measurement result will be returned to the Decision Mechanism for the next decision. When initializing those hash values, Measurement Mechanism needs to compute the measured values of all objects that need to be measured and store these values in the TF card through Primary Trust Base.

4.1.4. Decision Mechanism. The main task of the Decision Mechanism is to support the composite measurement which needs to consider multiple measurement results. In contrast, judging based only on the hash value of the program is the basic measurement because the decision strategy is simple. We only want to verify the feasibility of TSB based on a TF card on commercial Android devices, so we adopted a simple decision strategy. The Decision Mechanism obtain decision rules from the Baseline Repository and determines the behavior of Control Mechanism by comparing the result from Measurement Mechanism with the related measured value in the TF card.

4.1.5. Baseline Repository. Baseline Repository provides TSB baseline information, which can be in the form of configuration information required by the TSB, including measurement plans, control policies, and decision criteria, or in the form of results output by TSB, such as measured values. In our design, the correct measured values are safely stored in the TF card and the current ones are stored in the Baseline Repository.

4.1.6. Underpinning Mechanism. Underpinning mechanism connects TSB with the outside world and helps them interact with each other. The device obtains the trust support of TSB through this mechanism. This mechanism is also responsible for the management and configuration of the trust strategy. However, the management of the trust strategy is not the focus of this paper, so there is no related display in Figure 3.

4.2. Trusted Boot. The trusted boot is an essential capability of the TSB we designed. In order to cooperate with the TSB as an application, we implement the TSB to the trusted boot as a retrospective measurement, as shown in Figure 4. We have modified the star style trust measurement model. After TSB is confirmed to be trusted, it becomes the central node of star style trust measurement structure. The measurements of any trusted boot cannot be valid unless the TF card is binding to a trusted, local RTM because an adversary may simply insert the TF card into an untrusted device and replay any wanted measurement sequence, that is, create arbitrary PCR values similar to a TPM reset attack [33, 34]. Since the TF card is not physically bound with the local RTM, we need the TF card to authenticate the RTM to ensure that it is a trusted code. In addition, we make the TSB turn off the device when the TF card is detached in case of device running without TF card.

Our TSB is designed in the form of an Android application to avoid modified firmware. So, it is necessary to wait for the device to finish booting before the TSB launches and performs retrospective measurement. In order to ensure the timeliness and validity of TSB measurement, there are two points to be ensured: (1) TSB starts immediately after the device boot and (2) TSB measurement operation is not allowed to be interrupted by other programs.

In order to make TSB run immediately after the device boot, the TSB registers a broadcast receiver to receive Android's boot broadcast. After receiving that broadcast, TSB's measurement activity starts immediately. In addition, the priority of the broadcast receiver is set to maximum value 1000 to ensure TSB to be the first to start.

The measurement operation must ensure effectiveness and atomicity so as not to be disturbed by user behavior or other processes. We set the TSB as the Android Device Administration application to implement the kiosk mode which is the lock task mode that allows Android devices to run tasks in an immersive, kiosk manner. When the device runs in lock task mode, users without permission cannot receive notifications, access unauthorized applications, or return to the home screen. We lock the screen in the measured activity, so the user cannot exit or perform any operations until the measure boot process ends.

In Figure 4, the trusted boot process of an Android device with our TSB is divided into the following parts:

(1) After the power button is pressed, the bootloader will be loaded into the internal RAM (Random Access Memory). Then, the bootloader will initialize the memory and load the kernel. After the kernel starts, the kernel sets up the interrupt controller, memory protection, cache, and process scheduling and then

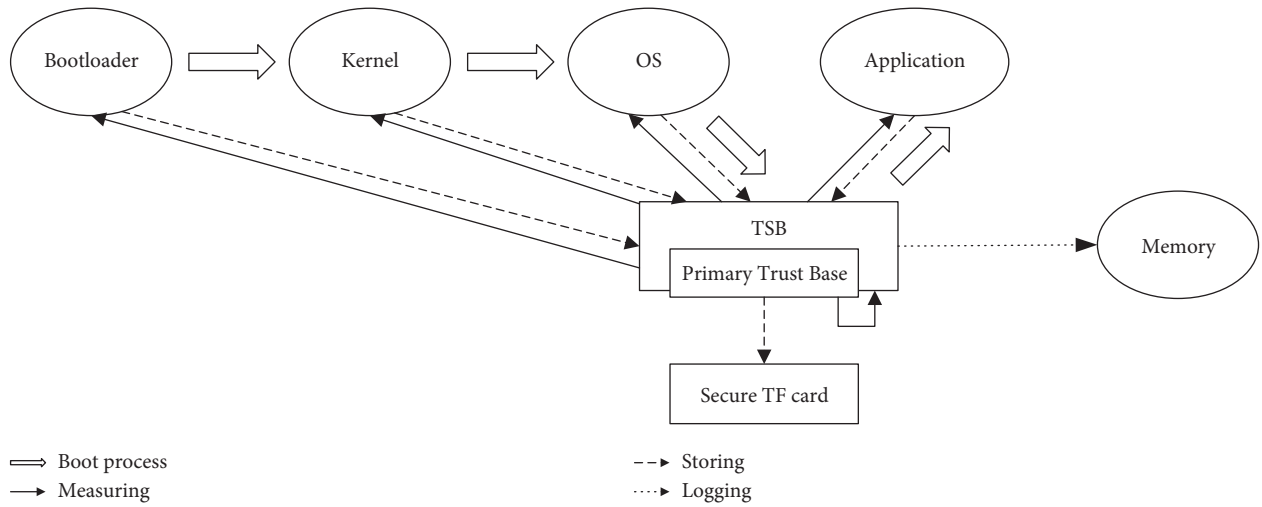

Figure 4: Trusted boot.

starts the user space process. After that, the init process starts, including parsing the init.rc script, loading the file system, starting zygote, and starting the system process. Zygote sets the Java runtime and inits memory for Android objects. Finally, the system service starts.

(2) After the system services start, the Primary Trust Base receives the system boot broadcast, skips the system interface launcher, and directly starts the measurement of TSB by using the secure TF card. Before that, the TF card will confirm the identity of the Primary Trust Base by verifying the signature. The Primary Trust Base gets the measurement result from the TF card to decide to start TSB or shut down immediately.

(3) After the entire TSB becomes trusted and has the password to access the Baseline Repository, it performs retrospective measurement of the boot chain, including the bootloader, system kernel, and key files in the Android system. Considering the limited computing power of the TF card and the time it takes in the measurement, all the compute processes are completed by the TSB after the Primary Trust Base has confirmed that the TSB is trusted. The hash result still needs to be compared with the measured value in the TF card.

(4) After TSB has measured the boot chain, it exits the locked task mode and hands the device control to the user. The TSB starts a background process to monitor the launch and installation of applications in the system. TSB records each measurement result in the logs in the device storage represented by memory in Figure 4.

We marked the startup process in Figure 4 to ensure its clarity. It is worth noting that we only verified the feasibility of TSB, so there is no reporting operation in Figure 4.

Trusted boot can be formulated as the follows where i represents each entity on the boot chain. To perform measurement on TSB, the TF card performs the hash computation of it. After that, the TSB calculates the hash value of the concatenation of its hash value and each entity i. For the convenience of understanding, although there is no specific hardware form of PCR to store the calculated hash value in TF card, we still use it to mark the measured value in the equation.

$$\begin{aligned} \mathrm{PCR}_i &= \mathrm{HASH}\left(\mathrm{Value}_{\mathrm{TSB}}\right), \\ \mathrm{NewPCR}_i &= \mathrm{HASH}\left(\mathrm{PCR}_i \| \mathrm{NewValue}_i\right). \end{aligned} \tag{2}$$

After the trusted boot, TSB establishes a star style chain of trust, which covers the boot chain of Android devices. If any measurement fails to pass the Decision Mechanism, the Control Mechanism shuts down the device to achieve the effect of suspending the boot process. This provides a safe environment after the device boots.

4.3. Trusted Applications. Since the Android devices do not reboot frequently and the security threats to Android devices mainly come from malicious applications, we add a permanent process to measure user applications. A flexible time segment instead of fixed period is adopted for application measurement considering the user's sensitivity to timeliness and power consumption. That means TSB measures every time an application is opened or installed, not at a specified moment. As shown in Figure 4, in order to ensure that the TSB is trusted every time the application is measured, the TSB in advance needs to pass the trust verification by the Primary Trust Base and TF card. The trust strategy is simple because we only verify the feasibility of TSB. Only when the application is in the whitelist and has the same measured value as in the TF card can it be installed and launched.

TSB uses Android's accessibility service to monitor user clicks and obtain UI information to get the name of the application launched. Then, it measures the corresponding application file and compares it with the measured value in the TF card. The application launches normally if the

measurement result passes the Decision Mechanism, which is unaware to the users. If the integrity check fails, TSB will occupy the screen to warn the user and the launch will be discarded. As for the application installation, a service TSB finds the application name and file through the broadcast of the application installation obtained by a receiver. The Control Mechanism controls whether the application will be installed according to the measurement result.

Since the measurement is not a one-time, but a long-term process, two measures need to be taken in practice. (1) To prevent the service from being recycled, we use the Android foreground service to keep monitoring the application. The foreground service is hard to be killed by the system when the system memory is insufficient. (2) To deal with the service of being killed in some extreme cases, we set up a system-level alarm clock that broadcasts once a minute. We register a broadcast receiver for the alarm to check whether the foreground service is still running. If the TSB finds no such service in the system, the TSB turns it on. These two measures ensure that the measurement of user applications will always exist after the device is booted.

4.4. Security Analysis. As mentioned in Section 1, one of the guarantees of TPM's security capabilities is that it is used as a dedicated hardware for computing. On commercial devices with limited resources and cost, we achieve similar capabilities of TPM with TSB and a secure TF card, which is physically isolated from Android malware. In the face of potential attackers who control Android malware, our solution can protect users from adversary attacks from two aspects. The trusted boot process ensures the trust of the user's device use environment, so that the device is in a safe state every time it boots. Whether the application is installed or launches, it will be measured and verified by TSB, which also ensures that users will not mistakenly use malware carefully set by the attacker.

A typical trusted boot checks the integrity of the component before it runs. It can only guarantee the integrity of these components and the safety of the boot process. Our TSB also checks the integrity of all components in the boot chain of commercial Android devices without changing its hardware and firmware. Defense against attacks before measurement, such as attacks on the kernel, relies on some mechanisms of the operating system itself, such as enabling mandatory access control SELinux [35] and kernel control flow integrity (CFI).

In fact, the implementation cannot fully comply with the TCG specification. In an implementation of the trusted PC [36], the BIOS will start running after the system is powered on and then load the PMBR (Pre-MBR) in the TPM into the main memory of the platform. The security of the BIOS is also guaranteed by its own access control. We use dedicated hardware to ensure the safety of boot process to a certain extent but cannot fully realize the ability of trusted computing. As mentioned in the threat model, attacks on Android devices come from malware. The device is in a trusted state when TSB and secure TF card are installed for the first time. Then, TSB begins to continuously measure the application of the device to prevent malicious software attacks.

However, we cannot completely cover potential TF card replacement or software attacks against TF cards. For the former, it depends on whether the secure TF card manufacturer will give the TF card a unique identity, using a technology such as PUF (Physical Unclonable Function) [37]. For the latter, the TF card is connected via a bus, which leads to possible bus attacks [38]. However, for daily users, these situations are outside our attack model.

5. Experimental Evaluation

We conducted experiments on commercial android devices and TF cards. Unfortunately, since our goal is to solve the problem of trust extension for commercial Android devices, we are not able to compare with other solutions such as fTPM under the same hardware conditions.

We tested the feasibility of our design as well as the performance overhead. In fact, after verifying the feasibility, the time cost of the solution is an aspect that needs to be compared. The time overhead for details such as key generation depends on the speed of the products produced by the TF card manufacturer. In addition, in our solution, the time-consuming computing work is handed over to the TSB process for execution after it is measured. That is, the device hardware of the TSB used, such as the SoC of the Android device, is generally faster than discrete TPM in theory.

5.1. Implementation Details. We choose Xiaomi Mi Pad 4 LTE and the JW1292T8I-SDK TF card produced by Chengdu JAVEE Microelectronics as the experimental hardware. Mi Pad 4 has Qualcomm Snapdragon 660 AIE SoC and 4 GB of RAM, and its system version is Android 9.0.

The TF card contains the public key encryption engine, hash engine, and random number generator corresponding to TCG's TPM specification. It also has the ability of symmetric encryption, which was added in the TPM2.0 specification. Different from the TCG specification, the cryptographic algorithms used in the TF card belong to the Chinese commercial cryptographic algorithms. The security TF card used in our experiment contains an SoC built with a 32-bit RISC processor. This SoC has hardware implementation of SM2 [39], SM4 [40], and random number generator (RNG). The SM3 [41] hash algorithm is completed by the RISC processor. These functions have dedicated calling interfaces for our TSB.

Experiments have proved that TSB can complete the trust extension during the boot process and the measurement of application launch and installation while putting a small burden on performance.

5.2. Feasibility. We verify the feasibility of trust extension to the boot chain and application.

5.2.1. Boot with TSB and TF Card. During boot process of the device, the integrity of the critical files from the bootloader and kernel to the system and TSB itself must be

ensured. If the measurement result is inconsistent with the measured value in the TF card, the boot process will be terminated.

We replaced the kernel with a modified one and tested whether the device can boot. In addition, we tested the modification of key system files such as init.rc in the experiment. Most importantly, we also tested whether the device can boot normally when the TSB code is modified. As for the bootloader, we found that once it is modified, the device cannot boot or recover, so we mark the relevant column in boldface in Table 1. Table 1 shows the results, which verifies the effectiveness and feasibility of our TSB design to extend the trust in commercial device booting. Each case in the table is completely the same except for the modified part represented by ✓. The unmodified components are marked with •. The modified part should not meet the integrity measurement and cause the device to fail to boot up. It can be seen that any modification of any part of the boot chain can cause boot failure. The TSB can find the integrity damage in the boot chain and shut down the device.

Table 1: Cases of different modifications and their boot status.

	Bootloader	Kernel	System	TSB	Boot status
Case 1	✗	•	•	•	Boot up
Case 2	✗	✓	•	•	Shutdown
Case 3	✗	•	✓	•	Shutdown
Case 4	✗	•	•	✓	Shutdown

✓ stands for modified content. • stands for unchanged content. Since the bootloader of commercial device cannot be modified, it is marked with ✗. It can be seen that any cases that modify any content on the boot chain cannot be successfully started.

5.2.2. Application Trust Measurement

(1) Application Installation. Malicious applications are the main form of attack against commercial Android devices, so trust in applications is vital to users. We developed two applications called applications 1 and 2 and only added the first one to the Baseline Repository. We compiled and installed the code of the first app again after changing it and found that the installation failed.

As shown in Table 2, experiments prove that applications that are not in the whitelist or that do not match the measured value in it cannot be installed, even if these applications are not malicious, and their names and icons are the same.

Because the malware is not in the whitelist, it cannot be installed either. However, we still tested a few malware, called Spitmo, Tapsnake, and DroidDream, which are all Trojan. The experimental results in Table 3 show that none of them can be installed and therefore no attack can be carried out.

(2) Application Launch. In fact, according to previous experimental results, it is not feasible to install malware on a device. In order to test applications that are not in the whitelist, we temporarily closed the Measurement Mechanism and Control Mechanism for application installation. We tested two applications, contacts and retromusic. The contacts app is in the whitelist and the retromusic is not.

Experimental results in Table 4 show that TSB can prevent applications that are not in the whitelist from being launched. We did not test the modified applications in the whitelist because the previous experimental results have shown that such applications cannot be installed, let alone launched.

5.3. Performance Overhead. We also consider whether the impact of TSB on the performance and energy consumption of the device is within the range that users can tolerate. When designing the TSB, we adopted some imperceptible measures to users. When the application is installed and opened, the user will not get any feedback if the measurement passes, thereby improving the user experience and device response. In addition, after the Primary Trust Base measured the TSB, computing tasks such as encryption are no longer run by the TF card but are handed over to the device to increase the running speed.

We conducted tests on device booting, application launch, installation time, power consumption, and performance and counted the impact of TSB on the above aspects.

5.3.1. Time. We measure the impact of TSB on time from three aspects: device boot, application launch, and installation.

(1) Device Boot Time. Since many users do not use TF cards, we measured the average boot time of the device in the three cases of no TF card or TSB, only a secure TF card, and both TF card and TSB. The boot time interval is defined as the device screen lighting up until the user can operate it. Therefore, we obtain the time interval from when the device is started to when TSB hands over the control of the device to the user after the measurement.

Table 5 shows that the secure TF card has considerable influence on the boot time. It takes about extra 60 seconds for the system to boot with TF card only. In the experiment, we used a customized secure TF card but not a universal one. The cryptographic capability in the TF card needs to be initialized when the device boots up. The extra time comes from the initialization of the TF card and the system trying to mount it. However, the real time spent in measurement is within 20 seconds. Considering that ordinary users do not reboot their devices frequently in daily life, the increase in boot time has little impact on the daily use of users.

(2) Application Launch Time. Application launch is the most frequently used operation on Android devices. In order to test the impact of TSB on the launch time of the application, we selected 16 applications for experimentation, A1 to A16 in Figure 5. The 16 applications are Via browser, Facebook lite, Instagram lite, Spotify lite, Firefox lite, Google Contacts, Google News, Shazam, Speedtest, WhatsApp, Instagram, Google Maps, QQ Music, TikTok, YouTube, and Photoshop Express in descending order of size ranging from 1 MB to 140 MB. Their sizes are marked as orange dots in Figures 5

Table 2: Installation status of the application in the whitelist before and after modification and the application not in the whitelist.

Application	Measured value	In the whitelist	Installation status
Application 1	eba4e2dbe4c75bd1b592c0625e09d550727fb5a1edeacd771e5351316c016229	Yes	Installed
Modified app 1	a82d263869037a9e80e359bbe3e2b2c3130c562e32ca9165a2adaf8fc96df60e	Yes	Not installed
Application 2	ae9123de2fc403666c9f48a6546fdfc5257ff69536f7c4d8a8d5d327d6a4e061	No	Not installed

Table 3: Installation status of malware.

Malware	Measured value	In the whitelist	Installation status
Spitmo	c6fa34e54cc9b42ac33e39d59d0f063cf6e91dfd812ec6f56fc11c09a3f741a0	No	Not installed
Tapsnake	9df604b05082df6e1a891533f1f3cb32627730cf702be06ad5e3e5ed5ed3ea1c	No	Not installed
DroidDream	ec80b2d0256b0ac75a48a3698e0814a71b3cf5fe8c86d06776fe8f61b9937cdf	No	Not installed

Table 4: Launch status of applications in the whitelist and not in the whitelist.

Application	Measured value	In the whitelist	Launch status
Retromusic	aee539f03ca21386c3394c98b18a5eb322486662b9529b91422c5a70b9b57a64	Yes	Launched
Contacts	ad38108cd8c9526966762c4e2d7704b97c7a5f343a2eba2ac827647e028e3219	No	Not launched

Table 5: Average boot time in different cases.

Case	Time (s)
W/o TF card or TSB	42.37
W/TF card only	105.98
W/TF card & TSB	122.67

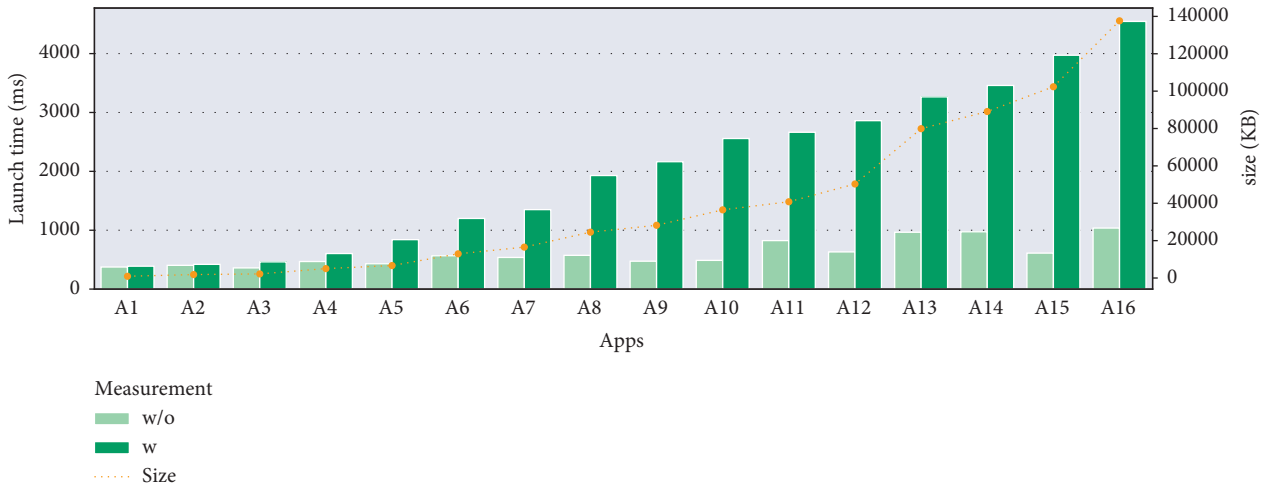

Figure 5: Application launch time.

and 6. We opened ten times and calculated the average time for each application with and without TSB and measured the time. TSB checks the integrity of the application while it is preparing to launch. Then, the application is ready to start until the check is complete. If the check fails, this launch will be discarded.

It is worth noting that to avoid interference from other factors as much as possible, we took some measures to ensure the consistency of the test environment, such as cleaning up the background before launching application and keeping the room temperature constant. In Figure 5, it can be seen that the application launch time is affected in the presence of TSB. The average delays shown are between 0.3 seconds and about 4 seconds. Since the main time consumption comes from the hash operation, the time required for measurement also increases as the application package size increases.

(3) Application Installation Time. We tested the installation time of those 16 applications and added them to the whitelist in advance to ensure that the applications can be installed. When TSB detects that the application is installed, it will first jump to its own activity and wait for the measurement to be completed before TSB jumps back to the installer.

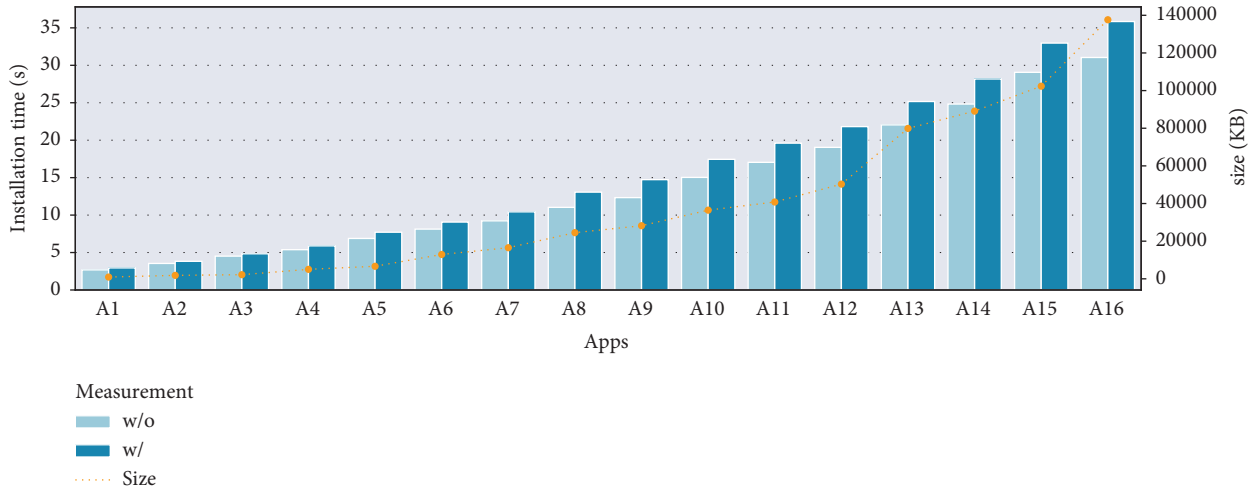

Figure 6: Application installation time.

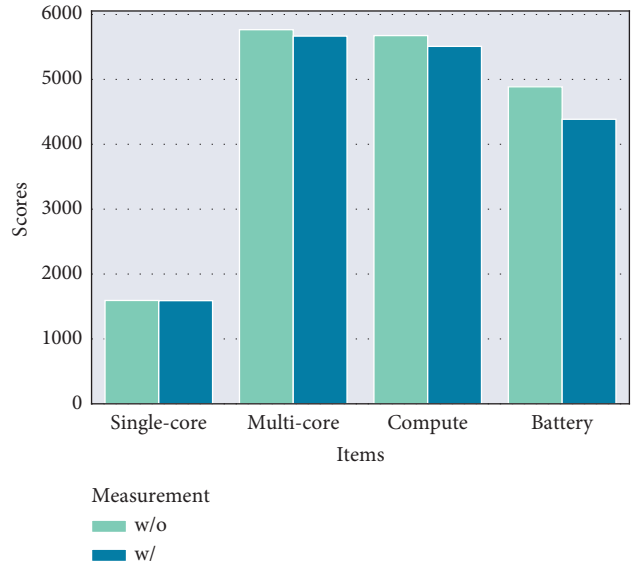

Figure 7: Performance and battery life.

Similarly, the increase in installation time is also caused by calculating the hash value of the applications. Figure 6 shows that the delay of application installation is at least less than 1 second and at most about 5 seconds. Moreover, for users, the frequency of application installation is low. From the experimental results, it can be considered that the measured installation has little impact on the user experience.

5.3.2. Performance and Battery Life. Since TSB spends most of its time on the measuring the application launch and installation, the trusted boot takes a small proportion of the time in the daily use of the device. Therefore, the impact of TSB on performance and energy consumption mainly comes from the always running foreground service. We use the mainstream benchmark application GeekBench 4.4.0 for testing. Figure 7 shows the results of the experiment. Single-core scores and multicore scores reflect the processor capabilities. The compute score here represents the score of GeekBench's compute test item in Figure 7. The battery scores represent the endurance of the device in high resource requirement situation.

It can be seen from the results that after using the secure TF card and TSB, the processor and compute scores drop very little. However, the battery life score dropped by about 10%, indicating that the foreground service still has some impact on the device's battery life.

6. Conclusion

Ensuring the trust of Android devices can reduce attacks on IoT and edge facilities from the mobile domain. This allows security research to focus more on IoT and edge devices themselves. Existing methods for measuring the trust of Android devices all need to modify the underlying firmware

or hardware of the device, or both. But it is unrealistic for most Android devices on the market, especially smartphones, to be promoted by device vendors. In view of this situation, we propose a TSB design for commercial Android devices. The device does not need to modify any hardware and firmware but only add a specific TF card. Based on the star style trust measurement model, we designed TSB to ensure the trusted environment for device boot and application running. While our method ensures the safety of the device, the performance and energy consumption are not significantly affected. Considering that our TSB is still a prototype system, the performance overhead and delay caused by it will become lower with the software optimization.

In future work, improvements and in-depth research can be carried out in the following aspects:

(1) The trust strategy in this paper is simple and cannot be updated. Adding a trust strategy management center without affecting system performance and managing trust strategy is worthy of study.

(2) Trusted Network Connection (TNC) [42] is needed if TSB connects to the Internet to update the Baseline Repository or report the measurement result.

(3) We have not considered the situation of multiple devices and TF cards. The access control of devices based on TF cards is worth being studied.

(4) TSB needs to be designed with better robustness and compatibility. For application measurement, TSB needs to test whether it will not crash on the device with mass applications. In addition, TSB needs to be compatible with a variety of Android devices and Android OS versions from 5.0 to 11.0, accounting for 94.1% of the current Android version distribution. As an application, TSB requires code protection, such as obfuscation techniques [43].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (no. U1936122) and Primary Research & Development Plan of Hubei Province (nos. 2020BAB101 and 2020BAA003).

References

[1] Y. Lee, S. Woo, J. Lee, Y. Song, H. Moon, and D. H. Lee, "Enhanced android app-repackaging attack on in-vehicle network," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 5650245, 13 pages, 2019.

[2] M. Park, J. Han, H. Oh, and K. Lee, "Threat assessment for android environment with connectivity to iot devices from the perspective of situational awareness," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 5121054, 14 pages, 2019.

[3] D. Wang, J. Ming, T. Chen, X. Zhang, and C. Wang, "Cracking iot device user account via brute-force attack to sms authentication code," *Proceedings of the First Workshop on Radical and Experiential Security*, pp. 57–60, 2018.

[4] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Dl-droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, Article ID 101663, 2020.

[5] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, Article ID 8863617, 11 pages, 2020.

[6] Y. Pan, X. Ge, C. Fang, and Y. Fan, "A systematic literature review of android malware detection using static analysis," *IEEE Access*, vol. 8, pp. 116363–116379, 2020.

[7] S. Arshad, M. A. Shah, A. Khan, and M. Ahmed, "Android malware detection & protection: a survey," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 463–475, 2016.

[8] D. Fu and Y. Liu, "Remote attestation on behavioral traces for crowd quality control based on trusted platform module," *Security and Communication Networks*, vol. 2021, Article ID 8859618, 12 pages, 2021.

[9] C. Shen, H. Zhang, H. Wang et al., "Research on trusted computing and its development," *Science China Information Sciences*, vol. 53, no. 3, pp. 405–433, 2010.

[10] F. Dengguo, Q. Yu, W. Dan, and C. Xiaobo, "Research on trusted computing technology," *Journal of Computer Research and Development*, vol. 48, no. 8, p. 1332, 2011.

[11] Microsoft The Windows Team, "Update on windows 11 minimum system requirements," 2021, https://blogs.windows.com/windows-insider/2021/06/28/update-on-windows-11-minimum-system-requirements/June 2021.

[12] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "The untapped potential of trusted execution environments on mobile devices," *IEEE Security & Privacy*, vol. 12, no. 4, pp. 29–37, 2014.

[13] B. McGillion, T. Dettenborn, T. Nyman, and N. Asokan, "Open-TEE -- an open virtual trusted execution environment," *2015 IEEE Trustcom/BigDataSE/ISPA*, IEEE, vol. 1, pp. 400–407, 2015.

[14] H. Raj, S. Saroiu, A. Wolman et al., "fTPM: a software-only implementation of a TPM chip," in *Proceedings of the 25th USENIX Security Symposium*, pp. 841–856, Austin, TX, USA, August 2016.

[15] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "Armageddon: cache attacks on mobile devices," in *Proceedings of the 25th USENIX Security Symposium*, pp. 549–564, Austin, TX, USA, August 2016.

[16] A. Machiry, E. Gustafson, C. Spensky et al., "Boomerang: exploiting the semantic gap in trusted execution environments," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2017.

[17] Z. Ning and F. Zhang, "Understanding the security of arm debugging features," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, pp. 602–619, San Francisco, CA, USA, May 2019.

[18] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management," in *Proceedings of the 26th USENIX Security*

Symposium, pp. 1057–1074, Vancouver, BC, Canada, August 2017.

[19] C. Cimpanu, "Trust issues: exploiting TrustZone TEEs," 2021, https://www.bleepingcomputer.com/news/security/trustzone-downgrade-attack-opens-android-devices-to-old-vulnerabilities/.

[20] W. Shi, "On design of a trusted software base with support of tpcm," in *Proceedings of the International conference on trusted systems*, pp. 1–15, Beijing, China, December 2009.

[21] Trusted Computing Group, "TCG specification architecture overview," 2007, https://trustedcomputinggroup.org/resource/tcg-architecture-overview-version-1-4/2007.

[22] C. Shen, H. Zhang, D. Feng, Z. Cao, and J. Huang, "Survey of information security," *Science in China - Series F: Information Sciences*, vol. 50, no. 3, pp. 273–298, 2007.

[23] B. Zhao, H.-G. Zhang, J. Li, L. Chen, and S. Wen, "The system architecture and security structure of trusted PDA," *Chinese Journal of Computers*, vol. 33, no. 1, pp. 82–92, 2010.

[24] Trusted Computing Group, "TCG mobile trusted module specification," 2021, https://trustedcomputinggroup.org/resource/mobile-phone-work-group-mobile-trusted-module-specification/.

[25] Trusted Computing Group, *Tpm 2.0 Library Specification*, https://trustedcomputinggroup.org/resource/tpm-library-specification/, 2021.

[26] Trusted Computing Group, "Tpm 2.0 mobile reference architecture specification," 2021, https://trustedcomputinggroup.org/resource/tpm-2-0-mobile-reference-architecture-specification/.

[27] D. Chakraborty, L. Hanzlik, and S. Bugiel, "simTPM: user-centric TPM for mobile devices," in *Proceedings of the 28th USENIX Security Symposium*, pp. 533–550, Austin, TX, USA, August 2019.

[28] Trusted Computing Group, "TCG TSS 2.0 overview and common structures specification," 2021, https://trustedcomputinggroup.org/resource/tss-overview-common-structures-specification/.

[29] A. Bianchi, J. Corbetta, L. Invernizzi, Y. Fratantonio, C. Kruegel, and G. Vigna, "What the app is that? deception and countermeasures in the android user interface," in *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, pp. 931–948, IEEE, San Jose, CA, USA, May 2015.

[30] J. Reardon, Á. Feal, P. Wijesekera, A. E. B. On, N. Vallina-Rodriguez, and S. Egelman, "50 ways to leak your data: an exploration of apps' circumvention of the android permissions system," in *Proceedings of the 28th USENIX Security Symposium*, pp. 603–620, Austin, TX, USA, August 2019.

[31] C. Ren, Y. Zhang, H. Xue, T. Wei, and P. Liu, "Towards discovering and understanding task hijacking in android," in *Proceedings of the 24th USENIX Security Symposium*, pp. 945–959, San Diego, CA, USA, February 2015.

[32] D. Lie, C. A. Thekkath, and M. Horowitz, "Implementing an untrusted operating system on trusted hardware," *ACM SIGOPS - Operating Systems Review*, vol. 37, no. 5, pp. 178–192, 2003.

[33] S. Han, W. Shin, J. H. Park, and H. Kim, "A bad dream: subverting trusted platform module while you are sleeping," in *Proceedings of the 27th USENIX Security Symposium*, pp. 1229–1246, Baltimore, MD, USA, August 2018.

[34] N. Lawson, "Tpm hardware attacks," 2021, https://rdist.root.org/2007/07/16/tpm-hardware-attacks/.

[35] B. Radhika, N. N. Kumar, R. Shyamasundar, and P. Vyas, "Consistency analysis and flow secure enforcement of selinux policies," *Computers & Security*, vol. 94, Article ID 101816, 2020.

[36] Y. Fajiang and Z. Huanguo, "Design and implementation of a bootstrap trust chain," *Wuhan University Journal of Natural Sciences*, vol. 11, no. 6, pp. 1449–1452, 2006.

[37] A. Cui, C. H. Chang, W. Zhou, and Y. Zheng, "A new puf based lock and key solution for secure in-field testing of cryptographic chips," *IEEE Transactions on emerging topics in computing*, vol. 9, no. 2, pp. 1095–1105, 2019.

[38] J. Boone, "TPM Genie," 2021, https://www.nccgroup.com/ae/our-research/tpm-genie/.

[39] International Organization for Standardization, *IT Security Techniques — Digital Signatures with Appendix — Part 3: Discrete Logarithm Based Mechanisms*, International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 14888-3:2018, 2018.

[40] International Organization for Standardization, *Information Technology — Security Techniques — Encryption Algorithms — Part 3: Block Ciphers*, International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 18033-3:2010, 2010.

[41] International Organization for Standardization, *IT Security Techniques — Hash-Functions — Part 3: Dedicated Hash-Functions*, International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 10118-3:2018, 2018.

[42] H.-G. Zhang, L. Chen, and L.-Q. Zhang, "Research on trusted network connection," *Chinese Journal of Computers*, vol. 33, no. 4, pp. 706–717, 2010.

[43] S. Dong, M. Li, W. Diao et al., "Understanding android obfuscation techniques: a large-scale investigation in the wild," in *Proceedings of the International Conference on Security and Privacy in Communication Systems*, pp. 172–192, Springer, Singapore, August 2018.