Proceedings of the 7th International Conference on Computing Methodologies and Communication (ICCMC-2023)
IEEE Xplore Part Number: CFP23K25-ART; ISBN: 978-1-6654-6408-6

Secure Mobile ID Architecture on Android Devices based on Trust Zone

Kapil Kant Kamal
Centre for Development of Advanced Computing (C-DAC),
Mumbai, India
kapilkantkamal@gmail.com

Sunil Gupta
University of Petroleum and Energy Studies (UPES),
Dehradun, India.
s.gupta@ddn.upes.ac.in

Padmaja Joshi
Centre for Development of Advanced Computing (C-DAC),
Mumbai, India.
padmaja.cdac@gmail.com

Monit Kapoor
Chitkara University Institute of Engineering and Technology, Chitkara University,
Punjab, India
monit.kapoor@chitkara.edu.in

2023 7th International Conference on Computing Methodologies and Communication (ICCMC) | 978-1-6654-6408-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCMC56507.2023.10083903

***Abstract*— Owing to their mobility, persistent connectivity, and variety of applications, cell phones are now more often used as a primary computing platform than desktop computers and laptops. Users have widely adopted mobile platforms to perform sensitive tasks ranging from personal online payments to remote access to organizational services. Mobile devices can store a lot of data, including sensitive data like login credentials, photos, videos, personal information, etc. Through attacks on Mobile devices, attackers stole users sensitive and personal data which has become a critical issue nowadays. This article investigates the security of Android storage models and investigated Trusted Execution Environment (TEE) as a potential countermeasure for such attacks, and discuss a new approach to mobile identity (Mobile ID) based on TrustZone. Further, this study undertakes a design consideration analysis from the standpoint of system vulnerabilities.**

Keywords*— *Software Vulnerabilities; Trusted Execution Environment (TEE); Android Application; Mobile System Security

I. INTRODUCTION

Technological development in the past decade has led to a sharp increase in the use of mobile platforms like smartphones. Users become increasingly at ease leveraging their phones for online transactions, games, entertainment, and shopping via mobile applications. As a result, mobile platform manufacturers and application developers have been pushed to invest in the security of their platforms and applications [1, 2]. Trusted Execution Environments (TEEs) are a crucial component of mobile device security architecture. They offer a context for execution where security-critical applications can run separately from the Rich Operating System, such as user authentication, mobile payment, and digital rights management. Security has been a priority for both Android and IOS Operating Systems (OS), the most popular mobile OS. TEEs are isolated from the rest of the system using hardware primitives. Moreover, a robust security architecture for this environment is now essential, particularly considering Secure Application Execution (SAE). Mobile device security is mostly built on the "Advanced RISC Machine (ARM)" TrustZone technology, which is commonly used to offer trusted execution environments for security-sensitive services [3, 4].

For TEEs, the following two conditions must be met:

1) **Memory Isolation:** Memory isolation is the protection of TEEs' address spaces from a kernel. There are software-only methods like Virtual Ghost and hardware-based methods like ARM TrustZone and Intel SGX. Software-based methods use kernel deprivileging or compiler instrumentation to kernel memory and TEE memory should be kept apart. [5].

2) **Attestation:** Attestation is a means to check if the application is running in a TEE environment. A chain of trust must be developed within the system to provide proof of attestation. As soon as the system boots, cryptographic hash chains are often used to create the chain of trust. Therefore, for accurate attestation, it is crucial to establish the root key of a hash chain (also known as the root-of-trust, or RoT) securely [5].

Our research focuses on certain versions of TEE and, in detail, the following contributions are made:

- A review of the design and description of TEE is presented along with static and dynamic analysis and a thorough explanation of their links and integration with the Android system.
- Numerous design faults are identified in the decryption and verification processes used to load private Trusted Applications. Further, these defects are considered as an advantage to recover all plain text data from the device. This study is performed by considering the TEE's runtime Trusted Application loader.
- At the user layer, TEE provides trusted applications with security such as Mobile based authentication, as well as trusted applications that provide services to the normal world. Cryptographic operations like encrypting, signing, and hashing are utilized to offer (Trusted Authorities) TAs with these functions. Sensitive information on mobile phones is encrypted using Cryptographic, which is also used to demonstrate the device's identification.
- A Mobile ID-based system, the brains behind all hardware-backed cryptography services are prototyped.

II. RELATED WORK

This section discusses additional efforts to create a trustworthy execution environment.

Normal and secure worlds are supported by ARM TrustZone. Data stored in the secure memory space is not accessible from normal world software it will access through specific applications. As a result, the underlying hardware technology for earlier TEE implementations was typically ARM TrustZone. In these earlier experiments, trusted software commonly referred to as an ARM TrustZone security monitor detects and prevents kernel accesses to trusted user files and data. In these earlier experiments,

trusted software commonly referred to as an ARM TrustZone security monitor detects and prevents kernel accesses to trusted user files and data. The architecture of ARM TrustZone-based TEEs comes first. They only work on ARM devices as a result. Additionally, the Trusted Computing Base (TCB) size of the entire system is bloated by the security monitor in the secure world [6].

Abundant research has been done on the topic of offering a TEE for mobile applications. Trusty, a safe OS, offers an Android TEE. The Trusty OS is physically and intellectually separated from the rest of the device even though it uses the same CPU as the Android OS. Trusty coexists alongside Android. Although separated, Trusty has complete access to the main CPU and memory of the gadget. As the Trusty is isolated, it is shielded against malicious software installed by the end-user or potential attackers [5]. These days' mobile devices store a lot of confidential information, and it becomes necessary to provide secure storage to prevent mobile devices from attacks. The use of TEE for secure storage in mobile devices is mostly used these days. However, many OSs are implemented in monolithic kernel architecture. The proposed paper focuses on microkernel architecture implementation and overcoming the short comes of monolithic architecture. At the user layer, "MicroTEE" offers both trusted programmes that offer services to the everyday world and trusted applications with security services for key management and cryptography. Usually, Crypto services are utilized to provide "Trusted Authorities (TAs)". These services use basic cryptographic functions like encryption, hashing, and signing [5].

This paper includes a detailed description of the MicroTEE architecture. Further, the authors also have a well-formed prototype implementation mechanism and performance analysis. Although the proposed work is good security measures can be enhanced a little more.

The analysis of safe key storage options for Android is provided by Tim Cooijmans et al. [7]. The writers of this paper examined the various safe key storage options offered by Android cell phones. Additionally, the authors give Android OS access to the Android Key Store, which can aid in device binding and protect against root attackers on devices running the Trust Zone TEE [6]. However, this systemcan be improved further.

K. Kostiainen et al. [8], describe TEE as a low-cost, open platform for certificate management with strong security. The cost of employing hardware-based security to manage credentials is too costly.

Moreover, the security of sensitive data on mobile devices solely through software is insufficient. It is possible to provide unified management of the entire life cycle, including the development of user credential information and the secure transfer of that information to mobile devices, by employing On-Board Credentials (OBC) [9,10].

U. Lee et al. [11] proposed SofTEE, a software framework to securely store user credentials [11]. SofTEE logically divides its CPU privilege mode into a normal mode and a secure mode using kernel deprivileging. SofTEE can support TEEs via software based on these two virtual CPU configurations. SofTEE assumes that hardware supporting attestation, like a TPM, has an integrated Root-of-Trust.

In short, all the examined studies either lack the provided security or their implementation is not available. Moreover, some of the proposed mechanisms do not fit well in the practical scenario. To overcome all such problems, a new scheme has been proposed.

III. PRELIMINARIES

This section presents various technologies that enable secure container instantiation. This section provides a brief overview of TEE and current technology, Android OS, Hardware Isolation, ARM Trust zone, and how it is utilized.

Android OS

Google created the operating system known as Android OS. It was originally made available in 2007. The Linux kernel on which the operating system is based has been altered to better suit a mobile OS. The source code for the Android OS and its packages is provided only when a new final version is made available. Continuous improvement is not open-sourced. C++ is also supported, while Java is the language of choice for most apps. OS services on Android are primarily written in C++ as opposed to apps. On their smartphones, anyone can utilize the Android operating system [6]. This indicates that a vast number of manufacturers provide Android phones. In light of safe key storage, a few Android OS features are important. The first is Android OS file storage.

- All apps and services that are running on the OS use the directory "/data" to store their data.
- Apps store their data in the directory "/data/data". Each app receives a directory that only the intended app can access.
- The SD card is mounted in the directory "/sdcard

Hardware-Enforced Isolation

It has been suggested to use hardware primitives to create a TEE that functions differently from a cryptographic co-processor. Utilizing platform hardware capabilities, the TEE isolates code execution from the rest of the system and defends data stored there from attacks by external software. As opposed to the rest of the system's code, which we refer to as untrusted code, the code found in the TEE is considered trusted.

ARM TrustZone Technology

A group of hardware additions makes up the ARM TrustZone technology. It is possible to divide the global physical memory into secure and insecure sections in several ways. This can be carried out, for instance, at each slave level, in the memory controller, or in a global module. Either the partition is hardwired or it is programmable. Any system may have unique requirements; however, all systems must be partitioned so that any unauthorized access to secure memory or devices results in an external core abort, which is a security breach [12,13]. It enables the use of secure world mode and normal world mode for the device. Each mode offers a different execution environment. There are also two categories for hardware and software resources like memory and peripherals. The software components that are most important for security are often run in Secure

World. It has access to every piece of hardware and software, even those found in everyday life. Only resources from the normal world's hardware and software are accessible. The monitor mode (refer to Figure 1) that TrustZone technology provides to the ARM processor oversees switching between the safe and non-secure worlds. The secure world includes the monitor mode.

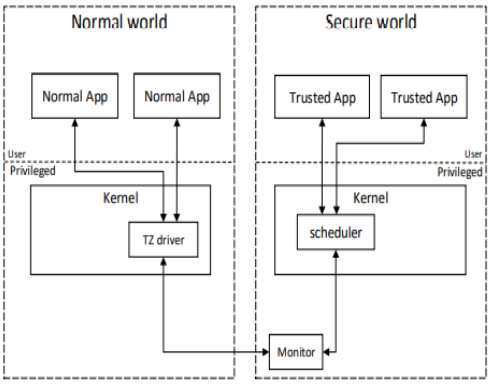

Figure 1: ARM TrustZone

Attacker Model

The attack model in this context may attack the android through any one or all cases.

- Malicious Application Attacker – In this scenario, the attacker attempts to exploit the stored private keys from a different application installed on the same device. The applications are assumed to be installed from an application market AppStore. It is assumed that the attacker has access to every permission that a program downloaded from such a store may ask for.
- Root Attacker - The attacker can examine the file system and run applications with root permissions. This model is an attacker who can execute an application with root access or utilizes root permissions on the device.
- Intercepting Root Attacker - This attacker captures user input that is entered by the user through the root permissions of devices.

IV. MOBILE ID UTILIZING TRUST ZONE

In this section, we proposed the extension of our previous work (m-ID) [14]. Mobile ID solution based on Public Key Infrastructure (PKI), which doesn't require any hardware. In this strategy, mobile information, which is distinct and will function as a Secure Element (SE) India-cured element, will contain user personal data such as demographics. A secure area will be built into the gadget itself to house the private data. This innovative design strategy ensures that the suggested solution will be autonomous, economical, secure, and interoperable.

In the proposed mobile ID solution, the user must first register on a mobile application where necessary demographic data is collected and verified using an Identity Issuing Authority (IIA). For instance, a human face will be taken into consideration for user authentication in a mobile device together with the demographic information recorded.

The Secord step generates Key Pair of users' data on mobile devices and keys stored inside the device such as Secure Element (refer to Figure 2). Keys should only be used for cryptographic operations once they are stored in the Keystore. The owner of the mobile device must continue to hold the private key to guarantee secrecy.

The third step involves sending a public key to the CA to obtain an X.509 certificate for use in online transactions and signing after the handheld device has finished creating key pairs.

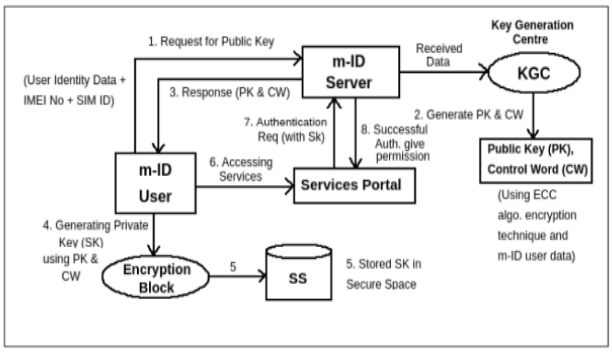

Figure 2: Secure Element of TEE

Now we provide the details of the steps in figure 2.

Step 1: The m-ID sends a request to the m-ID server.

Step 2: The m-ID server sends the received data to the Key Generation Center (KGC) and it generates the corresponding Public Key (PK) and Control Word (CW) using the ECC algorithm.

Step 3: The m-ID server sends the PK and CW to the m-ID user.

Step 4: m-ID user (android application) generates a private key (SK) using PK and CW.

Step 5: The generated SK is stored in the Secure Storage formed using TEE.

Step 6: After the key generation the m-ID user can send a request to access the service to the m-ID server through the portal.

Step 7: The authentication request is sent to the m-ID **server**

Step 8: The server responds with authentication approval.

List of use cases for which our proposal can be utilized are listed below:

- Sign documents such as PDF files.
- Secure online log-in and transactions.
- Secure e-commerce purchases
- Sign corporate transactions.
- Remotely access health records.

- E-Government: For usage on the e-government portal
- Tax payment and declarations.

Example Applications of TEE

Mobile payments (e-Commerce purchases) may be made via a peer-to-peer app, a merchant's app (in-app), the mobile browser, or at the point of sale (POS) of a business using a variety of communications technologies (e.g., Near Field Communication (NFC), Magnetic Secure Transmission (MST), Bluetooth & QR code) [15,16]. Mobile payments are becoming more and more popular among customers. NFC, QR codes, in-app, and in-browser payments are the new options. Payment credentials are saved and sent from one organization to the next in each of these cases. TEE can be of assistance in this situation and offer a better means of protecting such credentials both at rest and when being presented. Using TAs created expressly for each payment brand, TEE can offer a platform in mPOS payment acceptance devices for a trustworthy user experience and data processing. Hardware security can be used to encrypt data transfer from beginning to end. Users can create original applications on some mPOS platforms. Without using paper or plastic, value-added services like coupons, loyalty, and offers, can be enabled and securely tailored to users, resulting in a richer user experience [17].

In order to support the IoT, TEE is also used. IoT devices communicate the gathered sensor data to a central component of the public cloud because they typically have minimal processing power. TEEs can enhance the security assurances of the processing cloud environment because these data can be extremely sensitive [18].

Mobile identity user's identity may be credentials stored on the phone and associated with a PIN or fingerprint. Government entities may accept derived credentials stored on mobile devices to authenticate citizens and users. To access applications, login password, OTP, PIN, or biometrics may be required. In this context, mobile identity refers to a solution for managing user credentials. The TEE safeguards the application during its lifecycle and execution on the device, as well as user credentials, through hardware-backed storage and hardware isolation of credential processing. The TEE protects user interaction with the device by supporting fingerprint recognition, PIN entry, OTP display, and other authentication mechanisms [17].

Prototype Implementation and Primary Results:

We have implemented a prototype on the android simulator (android studio) and recorded primary findings regarding the file handling time. The m-ID server is created using an Apache web server and MySQL database is used. We have created a secure TEE on the android system and placed a simple application for storing typed text in the .txt file and then uploaded the file to the server using the procedure in Figure 2.

The obtained results are as in Figure 3. We have observed that the decent stable throughput is maintained by our proposal. However, as the file size increases the throughput slightly comes down.

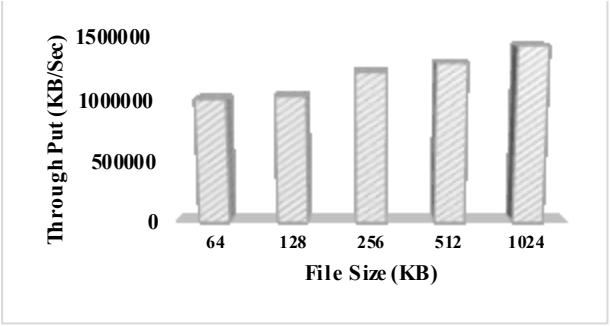

Figure 3: Throughput of Secure Element of TEE

V. SECURITY AND RELATED DISCUSSIONS

On Android, several security techniques can be applied to offer secure access to cryptographic primitives:

A. *Application-level Security*

The proposed enhancement framework has the following security mechanism at the application level.

1) **Protect application and user data**: When creating a mobile application, the focus should be given to security measures as well as usability and performance. Significant customer pleasure can result from a safe mobile application.

2) **Application-defined and user-granted permissions:** Mobile application permissions built upon security features can help the mobile device to support the data minimization, control, and transparency goals related to user privacy.

3) **Application Verification**: Ensure that applications being installed on the mobile device come from a valid and trusted source. Each mobile OS should have the ability to check the applications' digital signatures at the OS level.

4) **Anti-debugging:** Use to perform code obfuscation checks, hooking checks, emulator checks, and reverse engineering techniques.

B. *Framework level Security*

At the framework, we aim to provide the following security features.

1) **Protect system resources (including the network, camera, GPS, etc.):** Applications must grant explicit permission to use input devices (system resources) such as cameras, GPS, and microphones. A user must first expressly grant access authorization to a third-party application through the use of Mobile OS Permissions [14] for it to access these devices.

2) **Managing keys:** Enforcing security at the system level and proper management of keys is done.

3) **Secure Containers:** On a single, unified interface, work and personal profiles can be safely separated using a multi-user architecture.

4) **Device Resource Management:** Ability to enable/ disable device peripherals. Carrying out automatic, regular device integrity and compliance checks. Using device management software, need to be done.

C. *Kernel level Security*

To make the proposed TEE secure we provide the below-mentioned security features.

1) **System and Kernel Level Security:** Enable robust security at the OS by leveraging the security features in the underlying kernel [15].

2) **Secure Device Drivers:** Mobile Operating System locks down the processes which have access to vendor-specific device drivers to reduce the security impact of the modifications.

3) **Device Encryption:** Before accessing any data, users must submit credentials; this prevents all but the simplest activities from being carried out by the phone. User data should remain always encrypted.

4) **TEE:** A secure enclave located inside the main processor is known as a TEE. It operates simultaneously with the OS in a closed environment. It provides assurances of confidentiality and integrity for the code and data loaded into the TEE. TEE should be used for security-critical operations.

5) Secure Key Management: To make it more challenging to extract cryptographic keys from a device, store them in a container like a Secure Element or Hardware Security Module. Keys are utilized for cryptographic activities once they are in the Keystore, but the key material is not exportable.

VI. CONCLUSIONS

The paper discusses a new approach to mobile identity (Mobile ID). This work also suggests a method for employing secure elements to store sensitive data and keys insecurely on mobile devices. We discuss a novel approach for Mobile ID and the Numerous advantages offered by mobile ID. When using digital transformation, Mobile ID deployment is more cost-effective for deploying agencies and more convenient for users. These variables include ID management, user registration, the issuance and life cycle management of credentials, their use, and storage, and ultimately the use of m-ID for authentication, authorization, encryption, and signature in several scenarios. Deployment of Mobile ID should be seen as a means to an end a way for citizens to receive electronic services from the public or private sector or from another institution rather than the end in itself.

REFERENCES

[1] W. Enck, M. Ongtang, and P. McDaniel. Understanding Android security. IEEE Security & Privacy, 7(1):50 57, 2009.

[2] C. Miller, J. Honoro, and J. Mason. Security evaluation of Apple's iPhone. Independent Security Evaluators, 19, 2007.

[3] S. D. Yalew, G. Q. Maguire, S. Haridi, and M. Correia, "T2droid: A trust zone-based dynamic analyser for android applications," in 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, 2017, pp. 240–247.

[4] D. Hein, J. Winter, and A. Fitzek, "Secure block device–secure, flexible, and efficient data storage for arm trustzone systems," in 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1. IEEE, 2015, pp. 222–229.

[5] Azab, K. Swidowski, R. Bhutkar, J. Ma, W. Shen, R. Wang, and P. Ning, ''SKEE: A lightweight secure kernel-level execution environment for ARM,'' in Proc. Netw. Distrib. Syst. Secur. Symp.

[6] S. Zhao, Q. Zhang, Q. Yu, W. Feng, and D. Feng, ''SecTEE: A software-based approach to secure enclave architecture using TEE,'' in Proc. 26th ACM SIGSAC Conf. Comput. Commun. Secure. (CCS), London, U.K., 2019, pp. 1723–1740.

[7] Tim Cooijmans, Joeri de Ruiter,Erik Poll "Analysis of Secure Key Storage Solutions on Android" by , Erik Poll in Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices 2014.

[8] K. Kostiainen et al., "On-board credentials: an open credential platform for mobile devices," 2012.

[9] K. Kostiainen, N. Asokan, and A. Afanasyeva, "Towards user-friendly credential transfer on open credential platforms," in International conference on Applied cryptography and network security. Springer, 2011, pp. 395–412.

[10] Carlton Shepherd, Ghada Arfaoui, Iakovos Gurulian, Robert P. Lee, Konstantinos Markantonakis, Raja Naeem Akram, Damien Sauveron, Emmanuel Conchon: Secure and Trusted Execution: Past, Present, and Future - A Critical Review in the Context of the Internet of Things and Cyber- Physical Systems. Trustcom/BigDataSE/ ISPA 2016 : 168-177

[11] U. Lee and C. Park, "SofTEE: Software-Based Trusted Execution Environment for User Applications," in *IEEE Access*, vol. 8, pp. 121874-121888, 2020, doi: 10.1109/ACCESS.2020.3006703.

[12] Dongxu Ji, Qianying Zhang, Shijun Zhao, Zhiping Shi, Yong Guan MicroTEE: Designing TEE OS Based on the Microkernel Architecture 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering

[13] ARM Developer Module Available Online : https://developer.arm.com/documentation/ddi0333/h/programmer-s-model/secure-world-and-non-secure-world-operation-with-trustzone/how-the-secure-model-works [Accessed on 11-12-2022]

[14] K. Kamal, M. Kapoor, and P. Joshi, "Secure and flexible key protected identity framework for mobile devices,"International Journal of Information Security and Privacy (IJISP), vol. 16, pp. 1–17, 2022.

[15] Hussain, M., Zaidan, A.A., Zidan, B.B., Iqbal, S., Ahmed, M.M., Albahri, O.S. and Albahri, A.S., 2018. Conceptual framework for the security of mobile health applications on android platform. Telematics and Informatics, 35(5), pp.1335-1354.

[16] Caviglione, Luca, Wojciech Mazurczyk, Matteo Repetto, Andreas Schaffhauser, and Marco Zuppelli. "Kernel-level tracing for detecting stegomalware and covert channels in Linux environments." Computer Networks 191 (2021): 108010.

[17] Trusted Execution Environment (TEE) 101:A Primer 2018 by A SECURE TECHNOLOGY ALLIANCE https://www.securetechalliance.org/

[18] Gremaud, P., Durand, A., and Pasquier, J. (2017). "A secure, privacy-preserving IoT middleware using intel SGX", in Proceedings of the Seventh International Conference on the Internet of Things - IoT'17. p. 1–2. doi: 10.1145/3131542.3140258