Security Impact of Trusted Execution Environment in Rich Execution Environment Based Systems

** Jithu Philip*
*** Merin Raju*

Abstract

Security threats are growing in a very fast manner ever since the introduction and widespread use of mobile computing devices like smartphones became popular. So, there arises a necessity to introduce security mechanisms to deal with such threats in actual operating system environments. Trusted Execution Environment (TEE) is one such successful approach where dedicated secure hardware is used in combination with its own operating system software which works apart from the real execution environment for achieving an isolation from the real world processing. However, TEE still lacks a common design strategy as its implementation of is done by different manufacturers using their own hardware in a not so unified manner. So, here in this paper we try to study and follow the design strategies of a TEE with its basic concepts to analyze its security impact over a normal execution environment. As the use of mobile applications is growing day by day, the design strategies discussed in this document are mostly related and well suited for mobile platforms. Existing software based security mechanisms in mobile platforms like application sandbox is discussed in the later section of the document to analyze the type and the amount of vulnerabilities a TEE based system can fix over such strategies. The main application areas that a TEE can be securely employed is also discussed in the final section of this document to analyze the security impact that a TEE employed system can provide to a Rich Execution Environment.

Keywords : DRM, Kernel, Modular Programming, REE, Secure payment and authentication, TEE

I. INTRODUCTION

The wide spread use of a variety of digital media with highly sophisticated software, and its extensive use over the internet makes it highly vulnerable to attacks and introduced a possibility for intruders and hackers to manipulate the software's source code, thereby imposing attacks on the entire system. Any digital system which is compromised in such a manner seriously exposes the security of the system by affecting the confidentiality and integrity of its operations.

A. Need for a Hardware Based Isolated Execution Environment

There lies a higher demand to ensure security at its highest levels, especially in the case of multi-purpose mobile computing devices like smartphones. Because of its portable nature, most people make use of these devices as a storage place to keep their highly sensitive personal information. Apart from storing sensitive documents either offline (inside the device) or online (any cloud service that offers storage), the advances in processing capabilities makes these devices compete with any workstation kind of machines in almost all areas (except for the use of high end professional softwares).

So, people exploring highly sensitive details like banking credentials and personal user authentication details must need a higher security mechanism to keep these details safe in an encrypted storage as in most cases there is a need to permanently store it and to process it in an isolated manner in real time. So, a control in the information flow between the evolving entities needs to be achieved. If these principles are not followed while exploring the details, there is a possibility that an intruder taking control of an application can make use of flaws in the source code to take advantage of it, thereby compromising the security of the entire system [1].

Many software based security mechanisms, like

Manuscript Received : July 11, 2020 ; Revised : July 23, 2020 ; Accepted : July 25, 2020. Date of Publication : September 5, 2020.

*J. Philip is Multimedia Specialist with Philco Media, Kerala.
** M. Raju is Lecturer with Department of Commerce, Bishop Kurialacherry College for Women. Amalagiri, Kottayam, Kerala, India - 686 561. (email : merin.raju12s@gmail.com)

Doi : 10.17010/ijcs/2020/v5/i4-5/154785

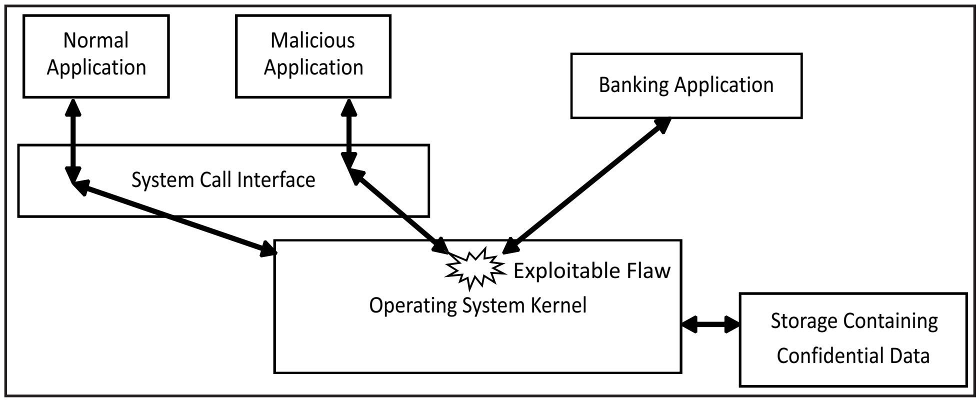

Fig. 1. Attacker Taking Control of the System Through Malicious Application

application sandbox, exist in operating systems to secure machines in Rich Execution Environments. As specified in a study related to Application Sandbox [2], it is proven that it is a better mechanism for most operating systems dealing with mobile platforms. It can work as a secure way of isolating one application from another and protect critical data and resources. However, due to lack of dedicated hardware and controlled software (one which is separate from the actual operating environment), the restriction of information flow between the evolving entities may not be possible in all cases. Even if most of the applications work in an isolated environment between themselves, there still arises the problem that an attack is still possible through a fake / malicious / unknown application which can exploit flaws in the real operating system, that is, a software based isolation may not be sufficient to hold attacks from intruders who exploit flaws in the actual operating system (Rich Operating System or Rich Execution Environment) which exposes the contents of the system.

A common approach an intruder may make use of to control the system by hacking an application is shown in Fig. 1. In cases like these, a dedicated hardware mechanism which is secure and is operated by its own secure software (like a separate Trusted OS) other than the actual Operating System is required.

Trusted Execution Environment (TEE) is one such approach with an isolated execution environment which makes use of kernel separation that enables it to provide end-to-end security by protecting the execution of authenticated code, confidentiality, system integrity, authenticity, privacy and data access rights. TEE's evolvement as a separate hardware along with dedicated secure software is growing as an essential need in most modern devices, especially in smartphones. Many device manufacturers are currently making this as a mandatory change, especially in their higher end devices. Real world implementations like Secure Enclave chips in iPhones, Titan M processors in Google Pixel, and the use of ARM processors with TrustZone configuration follow similar strategies.

B. Existing TEE Enabler Hardware Technologies

There exist different types of TEE enabler hardware implementations. One such kind of implementation uses CPU extensions where the processor is enabled with circuits that enables specific TEE enabling functionality. Some implementation like that includes the Arm TrustZone [9], Intel Software Guard Extensions [14], Intel System Management Mode (SMM) [15], and Sanctum [16]. There is another category which is implemented as a separate co-processor as part of the actual processor. Apple Secure Enclave Processor [3], [17], and Qualcomm Secure Processing Unit [18] are examples of these. The co-processors implemented in this way hold a dedicated non-volatile storage and RAM for reducing secured shared resource to be directly accessed by normal mode applications. This further secures the operations so that the resources that belong to the secured mode can only be accessed by programs running in secure mode.

There is another category of a TEE implementation that makes use of a separate processor other than the actual processor for secure operations. The Titan M [19] chip used by Google for Pixel smartphones (from Pixel 3 onwards) is one such category which enables tamper detection by default. The Trusted Platform Module [20] feature enables specific functions for trusted boot and remote attestation. The Windows Virtual Secure Mode (VSM) [21] as hypervisor enables two hierarchical privilege modes VTL0 (for normal world) and VTL1 (for secure world). AMD Secure Encrypted Virtualization (SEV) [22] is another technology that encrypts the virtual machine memory using hardware accelerated memory encryption.

The Intel Management Engine (ME) [23] is an autonomous subsystem built into the intel's processor chipsets. The ME is a firmware based on Minix OS that runs on a separate processor in Intel based systems with its own secure boot functionality.

II. DESIGN OF TRUSTED EXECUTION ENVIRONMENT

Trusted Execution Environment (TEE) is a secure area of the main processor which aims at the storage and execution of sensitive data in an isolated environment. Thus, the use of TEE ensures the confidentiality of data along with the integrity of applications being processed. The main advantage of using a TEE approach is that it provides better security for applications and data by executing them in an isolated environment, which offers protection against common software attacks imposed in Rich Operating System (Rich OS) or Rich Execution Environments (REE).

The ability to provide end-to-end security is achieved by the TEE's ability to offer safe execution of authorized security software, known as Trusted Applications (TAs). The major use of TEE is to protect the device and Trusted Applications (TAs) assets through its isolated execution environment.

The implementation of TEE is mostly done as a Dual Execution Environment where one is a non-secure environment (less secure environment where resources are publicly available to all applications based on request) and the other is a secure environment in which resources and data are isolated between applications.

So regarding the basics, the major implementation starts at hardware level by creating two environments that can run simultaneously on a single processor chip with a non-secure execution in one environment and a more secure, isolated execution in the other. The developers need to secure the system at the hardware stage as hacking of systems related to the lowest physical layer to tamper the boot process is also a known form of attack these days. The actual hardware design of a TEE based implementation may differ from those based on the manufacturers' criteria. So every TEE implementation does maintain different layers of isolated mechanisms to achieve a design strategy like this. The dual execution environment is a series of tasks executing in either the Rich Execution Environment or the Trusted Execution Environment with an intermediate environment that acts as a context switch which works in between either of these two execution environments.

The gateway, which is the TEE Entry and Exit environment that handles the task switch between either of these two execution environments usually works as a monitor mode. The execution path of tasks through different modes in a Dual Execution Environment is shown in Fig. 2.

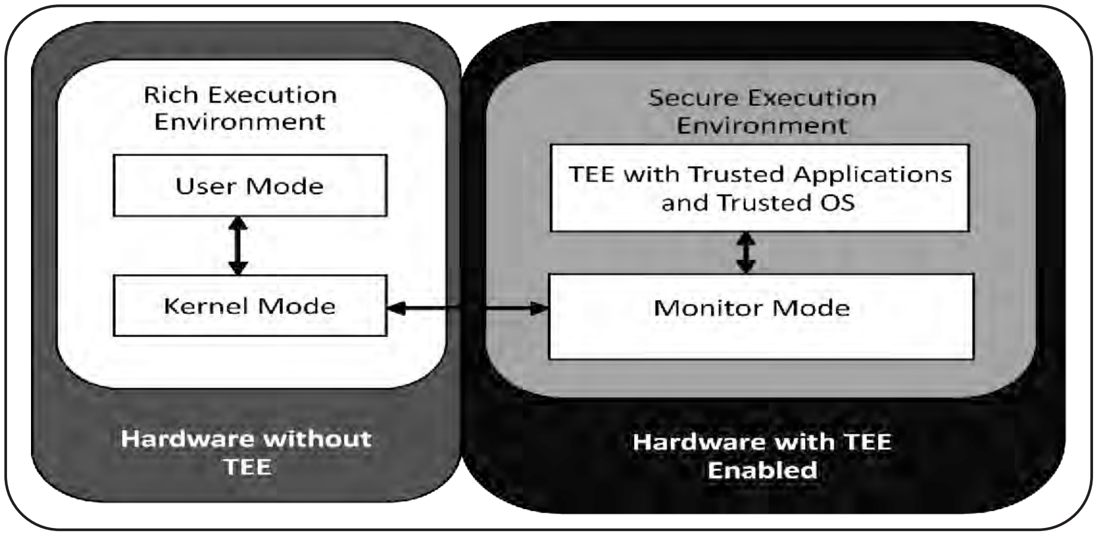
Rich Execution Environment
User Mode
Kernel Mode
Hardware without TEE
Secure Execution Environment
TEE with Trusted Applications and Trusted OS
Monitor Mode
Hardware with TEE Enabled

Fig. 2. Different Modes of Execution in a Dual Execution Environment

The Trusted Execution Environment (TEE) is normally a part of the main processor or can be implemented as a separate physical processor. As the major use cases of TEEs are related to smartphones and technologies involving mobile environments, different design strategies have been followed by different manufacturers. It is to be noted that, most of the existing TEE implementations still lack a common unified structure as the hardwares used in the implementation process are still very different by their own nature.

The most exclusive TEE like implementation strategy followed by popular manufacturers are :

Secure Enclave : Secure Enclave [3] is implemented as a part of the main processor chip in iPhones by Apple Inc., which resides as a coprocessor in iOS device's system on chip (SOC). It uses encrypted memory and does all cryptographic operations for Data Protection. During its hardware fabrication, each device's secure enclave is assigned a unique ID, which cannot be changed and the data that is written to the file system by Secure Enclave is further encrypted with a key bound with the UID and an anti-replay counter (which is stored in a dedicated nonvolatile Integrated Circuit(IC)). The processing of fingerprint and face data from Touch ID and Face ID sensors for user authentication is also done by the Secure Enclave.

Titan M is implemented as a dedicated separate chip which is separate from the actual processor of the device by Google in its Pixel devices [11], [12], [13].

Arm TrustZone : Another TEE system which relies on trusted hardware is the Arm TrustZone [9], [10] that has been implemented on the Arm processors (initially in Cortex-A series of application processors and was re-engineered for the new generation Arm microcontrollers, Cortex-M) and focuses on securing operations like user authentication,online banking, DRM etc.

III. SECURITY ARCHITECTURE OF TEE

As specified in section II, the design of TEE follows an uncommon strategy due to the varying nature of hardwares used by different manufacturers for its implementation process. The only facts that are common regarding the architecture are the general aspects.

The architecture of TEE must usually follow the execution of instructions to be done as a Dual Execution Environment, where one is a Rich Execution Environment and the other is the Trusted Execution Environment. Both of these environments are isolated from each other during their execution, and hence, there is no direct interception in between the execution. The security architecture of TEE is shown in Fig. 3.

The normal applications that require any services from

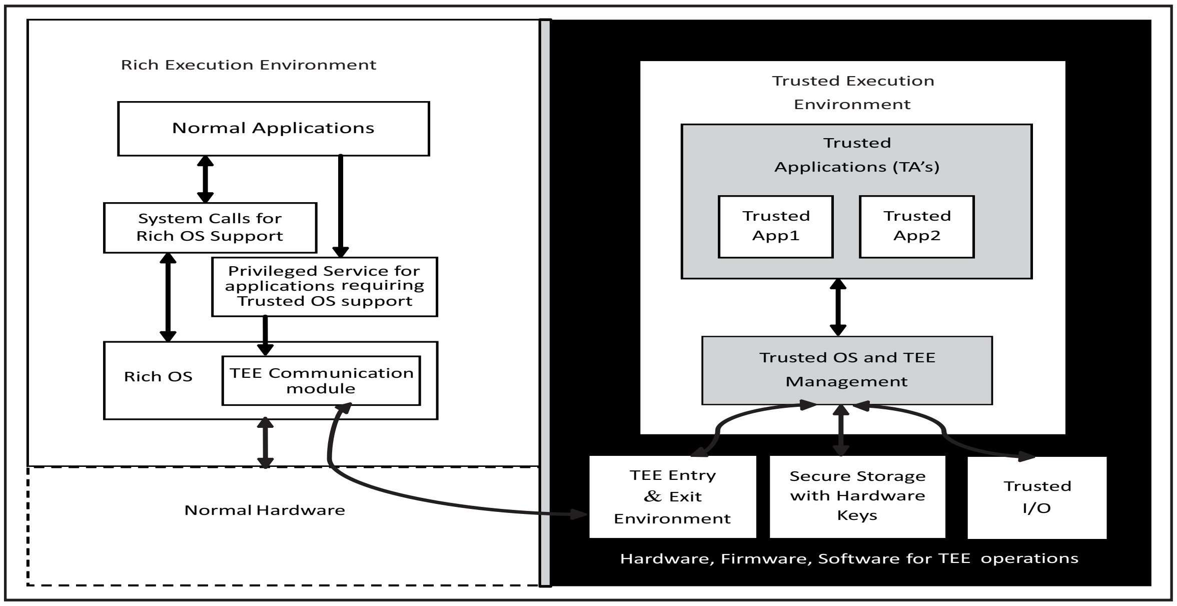
Rich Execution Environment
Normal Applications
System Calls for Rich OS Support
Privileged Service for applications requiring Trusted OS support
Rich OS
TEE Communication module
Normal Hardware
Trusted Execution Environment
Trusted Applications (TA's)
Trusted App1
Trusted App2
Trusted OS and TEE Management
TEE Entry & Exit Environment
Secure Storage with Hardware Keys
Trusted I/O
Hardware, Firmware, Software for TEE operations

Fig. 3. Security Architecture of Trusted Execution Environment

the devices' common hardware request such operations to the Rich OS through system calls. The hardware used for such operations are the normal processor, memory etc. which do the execution of common applications.

For the execution of applications requiring higher security, trusted hardware along with its trusted path is used. The applications executing on the hardware with TEE support are commonly termed as Trusted Applications. The higher demand for applications to be elevated and executed as trusted application is increasingly growing in many areas like in the case of banking applications, secure authentication and payment systems, Digital Rights Management (DRM) in multimedia related operations etc.

There are certain cases where a normal application is requiring services from Trusted OS. In such cases, it might request for elevated privileges (if verified and signed properly) through the TEE communication module. In most cases, there must be a TEE Entry & Exit module in between the two execution environments to do the authentication procedure, which acts as a monitor mode in between the two. The processor where Trusted Applications are executing can either be part of the actual processor or as a separate chip (see Section (I) for different TEE implementations by various manufacturers).

The TEE hardware can contain firmware that holds various instructions for TEE operations like code for signing the operating system boot loader (Secure Boot)) [3], hardware keys for authenticating the actual user and allowing access to the encrypted secure storage part. There can also be a Trusted I/O which is operated by a Trusted UI in Trusted apps. The content of TEE's secure storage is not static and can be securely updated.

The Trusted Kernel in a Trusted OS manages the TEE by making use of the design strategy of Separation Kernels, which was actually introduced many years ago and is described in the next section.

A. Separation Kernel

The separation kernel is one of the major design components used in the design phase of TEE. The separation kernel was initially introduced many years ago [5]. It acts as a security kernel [6] to simulate distributed systems. The major purpose of using a separation kernel is that it divides the system into different partitions and maintains a strong isolation between them. A strong definition regarding the separation kernel can be found on the Separation Kernel Protection Profile (SKPP) [7]. The SKPP defines separation kernel as "hardware and/or firmware and/or software mechanisms whose primary function is to establish, isolate, and control information flow between the maintained partitions".

The security requirements of designing a separation kernel are based on four main security policies [8] :

➭ **Data (spatial) Separation :** Data within one partition cannot be read or modified by other partitions.

➭ **Sanitization (temporal separation) :** Shared resource use is restricted. So the use of shared resource cannot be used to leak information between partitions.

➭ **Control of information flow :** Communication between partitions cannot occur unless explicitly permitted.

➭ **Fault isolation :** A security breach or vulnerability in one partition cannot be spread to other partitions.

The Trusted Execution Environment with a detailed view of kernel separation is shown in Fig. 4.

The Separation Kernel which acts as a basic building block of the Trusted Kernel consists of components for its two core operations :

(1) Secure Scheduling

(2) Information Flow Control / Inter-world Inter Process Communication (IPC) Manager

As described earlier, the separation kernel divides the system that is under our consideration into different partitions. To establish a proper workflow for each of the trusted applications which is executing in the TEE environment, applications are scheduled in such a way that there is a better isolation between the working entities (processes). The scheduler and IPC Manager in the separation kernel secures these operations, thereby accomplishing all of the security policies described above.

B. Modular Programming

As the major design component of TEE works with the concept of Kernel Separation (discussed in section A), the partitioned system along with its assets can be considered to be operating as separate modules. The advantage of decoupling application functionalities into independent modules is that it provides the entire TEE a higher level of

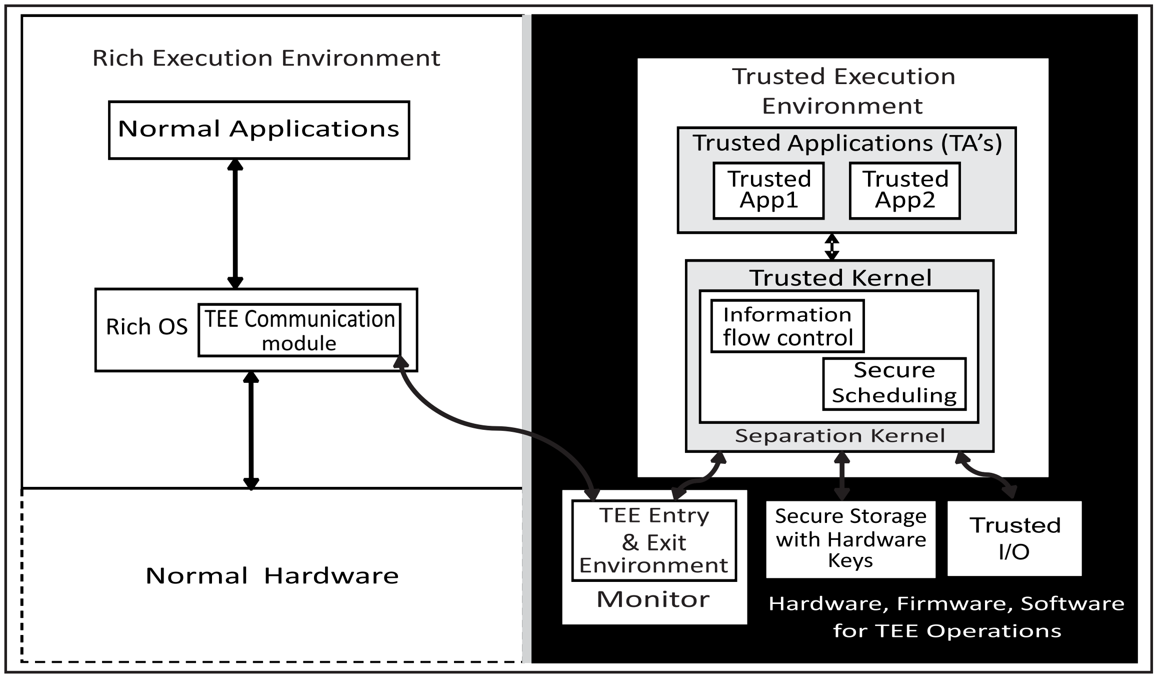
Rich Execution Environment
Normal Applications
Rich OS
TEE Communication module
Normal Hardware
Trusted Execution Environment
Trusted Applications (TA's)
Trusted App1
Trusted App2
Trusted Kernel
Information flow control
Secure Scheduling
Separation Kernel
TEE Entry & Exit Environment
Monitor
Secure Storage with Hardware Keys
Trusted I/O
Hardware, Firmware, Software for TEE Operations

Fig. 4. Trusted Execution Environment with Separation Kernel

reliability and security. So, it provides each individual module a higher level of fault isolation and prevents each module from creating a vulnerability for others.

For the secure communication between modules through data sharing, TEE makes use of secure payloads to be sent and received in between modules (by the use of mechanisms like objects serialization).

IV. RELATED SOFTWARE BASED DESIGN STRATEGIES IN MOBILE OPERATING SYSTEMS

As specified in the previous sections, TEE employs a hardware based execution environment, which makes use of a separate processing chip with its own dedicated software and/or firmware that creates a separate processing environment in conjunction with the normal operating system of the machine. So, as TEE holds a separate secure operating system for its operations, it is better to analyze some of the existing software based traditional access control mechanisms in mobile operating systems. One such security mechanism is the application sandbox [2], which is described in the next section.

Application Sandbox

Application sandbox is an access control technology which is mostly enforced at kernel level of the Rich OS. Sandboxing of applications is designed in such a way that users have the provision to choose what they share with an application. This allows users with the option that their critical data and access to the system itself is protected in its major share even if an application that is running in the system is compromised and is vulnerable to attack.

A survey regarding the application sandbox implementation in Android and iOS along with the increasing number of critical vulnerabilities found on both platforms are discussed [2], which gives a basic user awareness regarding the topic. The design of an application working in its sandbox directory in the mobile operating system iOS [3] is shown in Fig. 5.

So, based on researches and studies regarding this topic, it became known over the years that even if the Application Sandbox restricts an app to only deal with the allowed resources and data, it cannot be said that the technology is safe from all kinds of threats and attacks. One such attack is discussed in section I of this document (Fig.1.). It is the case where a user exploits flaws in the actual operating system from a malicious application, takes control of other apps through the actual operating system services and steals sensitive information.

So from this case, it is clear that an access control mechanism which is based on an operating system alone may not be sufficient in all cases. This is where a

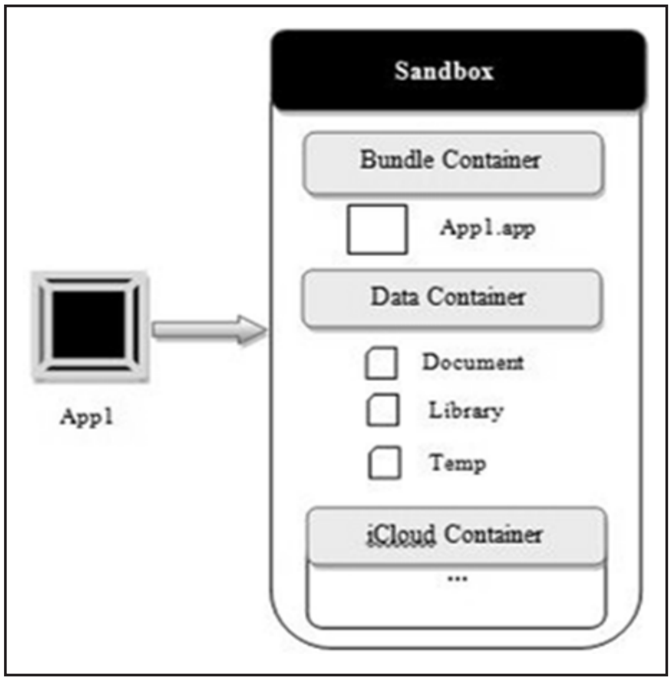

Fig. 5. An iOS Application Operating Within its Own Sandbox Directory

hardware based isolated execution environment other than the actual environment of the Rich OS is needed for achieving improved security. So, a TEE employment with its design strategies discussed in earlier sections can be an improvement for sessions requiring advanced security in such cases.

The major application areas, like a banking application where the user's sensitive details need to be processed securely can make use of technologies like this effectively. The efficient changes that a successfully employed TEE can make to such an application environment is discussed in the next section.

V. SECURITY ANALYSIS OF THE MAJOR APPLICATION AREAS THAT CAN USE TEE FOR IMPROVED SECURITY

A. Digital Rights Management (DRM) in Multimedia

Digital Rights Management (DRM) covers a set of access control technologies for protecting and restricting the use of copyrighted works. The use of DRM becomes widespread because of the fact that application areas requiring protection of copyrighted contents is growing in every aspects.

The major application areas making use of DRM policies are in the case of protection and distribution of copyrighted materials like, software and multimedia content. DRM also enables hardware locks in cases of proprietary hardware.

The general aspects followed in the distribution of the encrypted multimedia content along with the KDM holding the encrypted secret key of a Digital Cinema Package (DCP) [4] is shown in Fig. 6. It uses an asymmetric key pair (Two Level Encryption at the Sender's End) for the secure distribution of the multimedia content.

In this case, the multimedia content which is in raw format containing the actual video and audio streams is encoded to suitable format for representation. After successful encoding, the encoded content needs to be sent securely between the communicating entities. For that to occur, the encoded content is encrypted using a secret key, and the encrypted content is sent to the destination.

For achieving better security, the secret key itself is encrypted using the public key of an asymmetric key pair which is dependent on the sender and receiver. The encrypted secret key is also sent to the receiver. So, the receiver who holds the actual encrypted content along with the encrypted secret key can now decrypt the actual content.

For the decryption process to occur, the received encrypted secret key is decrypted first using the private key of the asymmetric key pair (which was used during the encryption stage). The private key of the asymmetric key pair is usually known to the receiver (usually hidden and securely stored in a tamper resistant chip at the receiver's end). In multimedia related operations, the corresponding public key of this private key is made public to the sender by the receiver for achieving transparency in operations. Usually, the secure storage of private key and the establishment of a public key related to that is done at the manufacturing or connection establishment step (whichever is required), as most of these are planned and executed at the initiation stage.

Once the encrypted secret key is decrypted, that secret key is used to decrypt the original content. So, in this case a two stage encryption is done at the sender's end. A more detailed description about the encoding and encryption of a Digital Cinema Package can be found on the work [4] which gives a reference about the needs of improving security while distributing multimedia contents.

As shown in the DCP encoding and encryption process

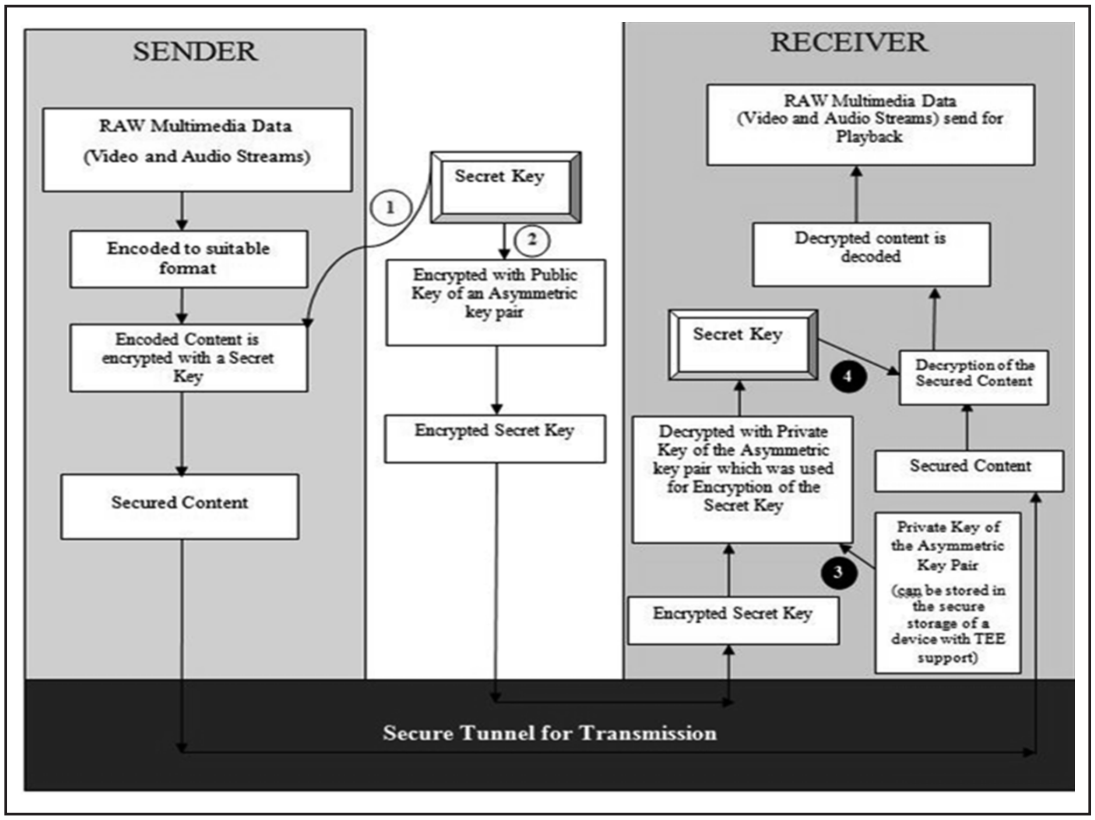
SENDER
RAW Multimedia Data (Video and Audio Streams)
Encoded to suitable format
Encoded Content is encrypted with a Secret Key
Secured Content
Secret Key
1
2
Encrypted with Public Key of an Asymmetric key pair
Encrypted Secret Key
RECEIVER
RAW Multimedia Data (Video and Audio Streams) send for Playback
Decrypted content is decoded
Secret Key
4
Decryption of the Secured Content
Decrypted with Private Key of the Asymmetric key pair which was used for Encryption of the Secret Key
Secured Content
3
Private Key of the Asymmetric Key Pair (can be stored in the secure storage of a device with TEE support)
Encrypted Secret Key
Secure Tunnel for Transmission

Fig. 6. The Distribution Model of the Encrypted Multimedia Content Along with a KDM Holding the Encrypted Secret Key of a Digital Cinema Package (DCP) with Basic Encryption and Decryption (shown in steps (1,2,3,4))

(Fig. 6.), the same concept can be used for the distribution of online multimedia contents. Nowadays, a vast majority of multimedia content is distributed to the user as a form of streaming service with the help of mobile applications. So in such scenarios, the same concept discussed above can be used along with the implementation of a TEE in the user side. If done so it can securely execute the user application, and safely store the private key of the asymmetric key pair which was used for the decryption of the secret key at the receiver's end.

B. Secure Payment and Authentication

The introduction of mobile applications made users with the facility to do banking through their smartphones and tablets, apart from accessing the banking websites all the time. So there arise the need to achieve a higher level of security, as highly sensitive user authentication patterns are to be executed in real time and the user data needs to be stored inside the system. TEE can be employed to secure a proper execution environment in these cases too. A common usage of this is where in place of a contactless payment that is employed in cases of mobile payment and digital wallet services like Apple Pay, Samsung Pay etc. the card information is digitized and stored inside the device itself. In cases like this, TEE is used in conjunction with Near Field Communication (NFC) for communication, and a Secure Element for user authentication details like Touch ID, Face ID is stored and crosschecked.

Apart from contactless payments, TEE can also be employed in executing and authenticating mobile commerce applications like mobile wallets, peer-to-peer payments, online banking applications etc. The real world communication of a user application between user, merchant, and bank with the support of a payment gateway, and which works in a TEE is shown in Fig. 7.

The user in this case shares the order and payment information with the merchant first. The merchant holding the order and payment details processes it and the payment details are forwarded to the payment gateway. The

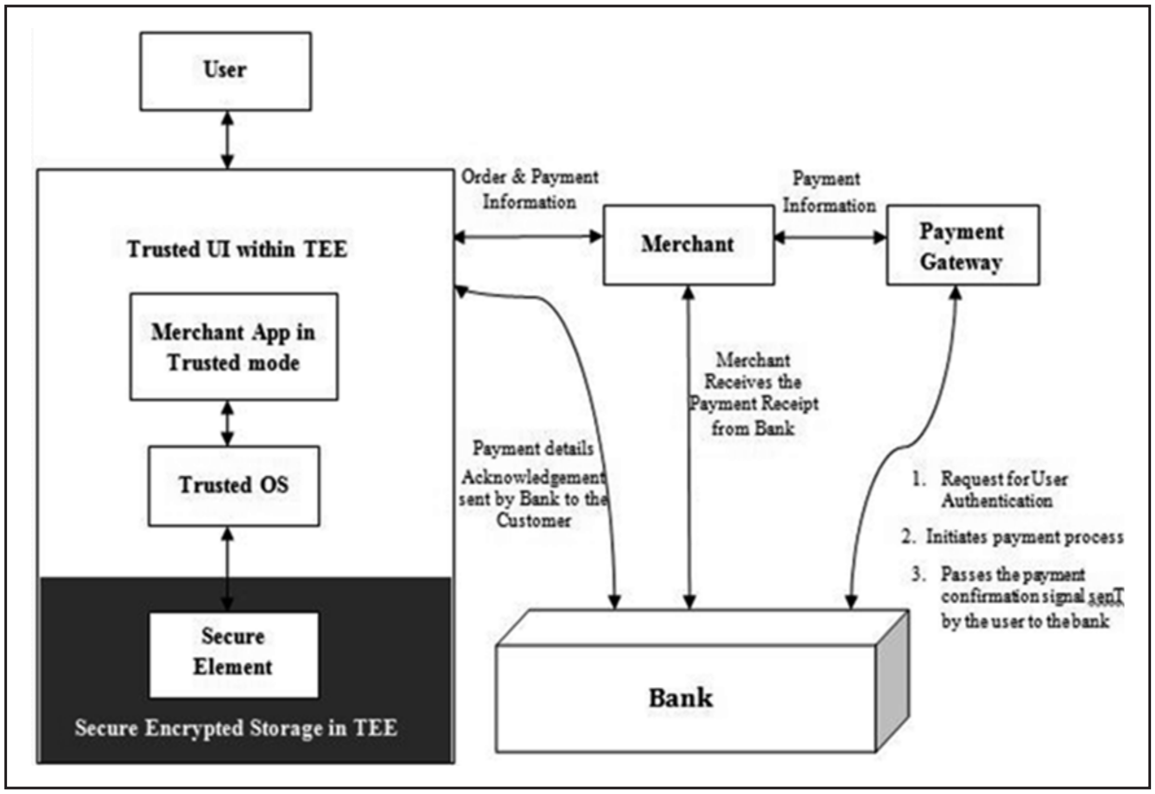
User
Trusted UI within TEE
Merchant App in Trusted mode
Trusted OS
Secure Element
Secure Encrypted Storage in TEE
Order & Payment Information
Merchant
Payment Information
Payment Gateway
Merchant Receives the Payment Receipt from Bank
Payment details Acknowledgement sent by Bank to the Customer
1. Request for User Authentication
2. Initiates payment process
3. Passes the payment confirmation signal senT by the user to the bank
Bank

Fig. 7. The Real World Communication of a User Application Between User, Merchant, and Bank With the Support of a Payment Gateway and Which Works in a TEE

payment gateway requests the bank to authenticate the user before continuing any further. The authentication to be completed is usually done by using a combination of factors, especially because the case involved is based on a mobile application. The use of a multi factor authentication can be done in this case. Once the authentication is complete, the payment process is initiated.

The bank may or may not acknowledge the user regarding this process. It is dependent on the type of payment method used, for example, in cases like internet banking, an OTP is sent to the user for confirmation. Once the user receives this, he can respond back to the gateway with the confirmation signal for processing the payment. If that is executed, the bank processes the payment, and once the payment is done, it sends an acknowledgement regarding the payment to the user and to the merchant. The payment gateway is closed on successful processing of payment and the user application is returned to the merchant's page which shows further details regarding the executed order like billing information etc.

The real world cases where the use of a TEE enabled device can produce a significant impact in the secure payment and authentication are discussed below :

Case 1 : User doing a contactless payment at a POS terminal

Consider an example, where the user visits a merchant and after purchase does the payment processing using a Point of Sale (POS) terminal as a contactless payment. Here in this case, normally the user's card based details are digitized and stored inside the smartphone. So with the support of TEE, the user makes a communication to the POS terminal using Near Field Communication (NFC). In this case, the authentication is usually done using a biometric ID which was previously stored inside the TEE secure storage. Most contactless payments usually does not make use of any other confirmations like OTP in majority of cases, and because of this reason, the transaction amounts per communication is set to low.

Here, in this case TEE can be implemented in the user's device which holds the digitized user's payment details for contactless payments.

Case 2 : User doing payment online

In this case, the user application works in conjunction with the merchant, payment gateway and the bank for a successful business transaction. Almost all of the business transactions are initiated by the user after interacting with the merchant and selecting the products he / she needs. After completing the transaction, payment needs to be done securely.

This is where the payment models are of varying natures these days. Because of the wide options available to the user, it can be done in many ways using mobile applications. For this to be done, the use of a TEE can be used for better authentication and secure payment.

So, the mobile application involved in this case, if elevated as a trusted application can make use of the TEE enabled hardware and trusted OS to work in an isolated environment which protects itself from other entities.

Here in this case TEE can be implemented in the user's device which executes the user application and securely updates all of the steps shown in Fig. 7.

The authentication factors used in these cases are different and are based on multiple factors with the use of authentication schemes like Multi-factor Authentication which is described in the next section.

➯ ***Multi-factor Authentication***

The authentication schemes employed while using mobile financial services and POS terminal may depend on multiple factors (multi-factor authentication) with different patterns to achieve better security. The combination of multiple factors may be like :

(1) Use of a physical card (Credit, Debit).

(2) A PIN or a Password.

(3) A biometric ID (Face ID, Fingerprint or Touch ID, Voice Authorization).

(4) Use of a specific network or GPS to identify location.

A two-step verification or two-step authentication is a commonly used form of authentication scheme to confirm a user's claimed identity by two factors. The two factors may include something that is known to the user such as a password and another factor that is unknown to the user which is to be authenticated and sent by a trusted party (like an OTP send over to the user by a bank through SMS).

For most contactless payments to occur, TEE is suited to store the Biometric ID related to the user. The process of authentication is as follows:

(1) A reference template (Face ID, Touch ID, Vocal Pattern) is scanned and stored initially while setting up the authentication.

(2) During the execution of application, the authentication validation is processed, as the user inputs the corresponding pattern (Face ID, Touch ID, Vocal Pattern) which was scanned and stored initially.

(3) A matching engine like a dedicated secure software is executed which matches the template with the new pattern which is scanned currently.

VI. CONCLUSION

The study of the basic TEE based design strategies allowed us to focus through different security aspects related to a dual execution environment approach. Since TEE works within a separate environment from the Rich Execution Environment, its implementation success gained a huge advantage over many of the major security flaws found in real world operating systems. The study specifies the advantages of TEE over other existing software based security mechanisms like Application Sandbox and because of its security advancements it is a better choice for users of modern mobile operating systems. The only factor that is not clear up to now is the uncommon design strategies followed by different manufacturers. As TEE based implementations makes use of both hardware and the software, its implementation details are mostly hidden or different based on the commercial types involved in these entities. Transparency regarding this issue is still possible and is under development as more common and unified approaches are happening in the developer's world. The discussion regarding the employment of TEE in major application areas allowed us to provide better transparency regarding this topic too. As the use of biometric based user authentication technologies are increasing year by year, we found it a necessity that a scope for betterment in advanced TEE based systems exists and its is needed in each and every consumer related device which requires secure user authentication.

ACKNOWLEDGEMENT

The authors thank anonymous reviewers for their valuable suggestions regarding the details of this document.

REFERENCES

[1] *"Introduction to Trusted Execution Environments,"* GlobalPlatform Inc., 2018. [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf

[2] J. Philip and M. Raju, "A formal overview of application sandbox in Android and iOS with the need to secure sandbox against increasing number of malware attacks," *Indian Journal of Computer Science, vol. 4, no.* 3, pp. 32 – 40, 2019.

[3] J. Philip and M. Raju, "An overview about the security architecture of the mobile operating system iOS", *Indian Journal of Computer Science, vol. 4,* no. 1, pp. 13–18, 2019. DOI: 10.17010/ijcs/2019/v4/i1/142412

[4] J. Philip and M. Raju "Encoding and encryption of digital cinema package," *Indian Journal of Computer Science, vol. 4,* no. 5, pp. 7–17, 2019. DOI: 10.17010/ijcs/2019/v4/i5/149455

[5] J. M. Rushby, "Design and verification of secure systems," *SIGOPS Oper. Syst. Rev., vol. 15,* no. 5, pp. 12–21, 1981. DOI: https://doi.org/10.1145/1067627.806586

[6] J. Ames, Stanley R., M. Gasser, and R. R. Schell, "Security kernel design and implementation: An introduction," *Computer, vol. 16,* no. 7, pp. 14–22, 1983. DOI: https://doi.org/10.1109/MC.1983.1654439

[7] "U.S. government protection profile for separation kernels in environments requiring high robustness," *Information Assurance Directorate,* June 29, 2007, version 1.03. [Online]. Available: https://www.commoncriteriaportal.org/files/ppfiles/pp_skpp_hr_v1.03.pdf

[8] M. Sabt, M. Achemlal, and A. Bouabdallah, " Trusted execution environment: What it is, and what it is not." *14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications,* Aug 2015, Helsinki, Finland. DOI: 10.1109/Trustcom.2015.357

[9] *Arm TrustZone Technology.* [Online]. Available: https://developer.arm.com/ip-products/security-ip/trustzone

[10] *Introduction to Trusted Execution Environment: ARM's TrustZone.* [Online]. Available: https://blog.quarkslab.com/introduction-to-trusted-execution-environment-arms-trustzone.html

[11] "The tiny chip that powers up pixel 3 security," *Wired.* [Online]. Available: https://www.wired.com/story/google-titan-m-security-chip-pixel-3/

[12] R. Triggs, *"Will Google's Titan M make it harder for the ROMing scene?," 2018.* [Online]. Available: https://www.androidauthority.com/titan-m-security-chip-915888/

[13] C. Hoffman, "Your smartphone has a special security chip. Here's how it works," *How-to Geek, 2018.* [Online]. Available: https://www.howtogeek.com/387934/your-smartphone-has-a-special-security-chip.-heres-how-it-works/

[14] Intel, *"Intel Software Guard Extensions," 2019.* [Online]. Available: https://software.intel.com/en-us/sgx/

[15] R. R. Collins, *"Intel's system management mode."* [Online]. Available: http://www.rcollins.org/ddj/Jan97/Jan97.html

[16] V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," in *USENIX Security Symposium.* USENIX Association, pp. 857–874, 2016.

[17] T. Mandt, M. Solnik, and D. Wang, "Demystifying the secure enclave processor," *Azimuth Security, 2016.* [Online]. Available: https://www.blackhat.com/docs/us-16/materials/us-16-Mandt-Demystifying-The-Secure-Enclave-Processor.pdf

[18] Qualcomm, *"Qualcomm Secure Processing Unit SPU230 Core Security Target Lite," 2019.* [Online]. Available: https://www.commoncriteriaportal.org/files/epfiles/1045b_pdf.pdf

[19] X. Xin, *"Titan M makes Pixel 3 our most secure phone yet," 2018.* [Online]. Available: https://www.blog.google/products/pixel/titan-m-makes-pixel-3-our-most-secure-phone-yet/

[20] T. C. Group, *“Trusted Platform Module (TPM),” 2018.* [Online]. Available: https://trustedcomputinggroup.org/workgroups/trusted-platform-module/

[21]*“Virtualization-based Security (VBS),” 2017.* [Online]. Available: https://docs.microsoft.com/en-:us/windows-hardware/design/device-experiences/oem-vbs

[22] AMD, *“AMD Secure Encrypted Virtualization (SEV),” 2019.* [Online]. Available: https://developer.amd.com/sev/

[23]*“Getting Started with Intel Active Management Technology (Intel AMT),” 2019.* [Online]. Available: https://software.intel.com/en-us/articles/gettingstarted-with-intel-active-management-technology-amt

About the Authors

Jithu Philip received M. Sc. degree in Computer Science from School of Computer Sciences, Mahatma Gandhi University, Kottayam, Kerala, India in 2014. He is currently working as Multimedia Specialist for Philco Media, Kottayam, Kerala, India. His research interests are in the areas of Operating Systems and Computer Security.

Merin Raju received M. Sc. degree in Computer Science from School of Computer Sciences, Mahatma Gandhi University, Kottayam, Kerala, India in 2014. She is currently working as Lecturer (Computer Science) with Department of Commerce, Bishop Kurialacherry College for Women, Amalagiri, Kottayam, Kerala, India. Her research interests are focused on Computer Security.