Intel® Trusted Edge Platform (TEP)

Design Guide

August 2022

Contents

Figures

Terminology

Table 1: Terminology

Term	Description
TEP	Intel® Trusted Edge Platform
BIOS	Boot software
Boot Guard	Secure boot is also known as boot-guard, any reference to boot-guard means secure boot
HWROT	Hardware Root of Trust
Intel® PTT	Intel® Platform Trust Technology. Firmware implementation of TPM2.0 spec.
iSecL	Intel Security Libraries
CSR	Certificate Signing Request
CS	Credential Service
AES	Advanced Encryption Standard
PKI	Public Key Infrastructure
AES-XTR	AES Counter Mode
SELinux	Security enhancement to Linux
BSP	Board Support Package
MAC	Mandatory Access Control
PKCS11	Public-Key Cryptography Standards

1.0 Introduction

1.1 Purpose

This document provides a high-level architecture and design overview of Intel® Trusted Edge Platform (TEP). TEP is intended to create a minimalistic, isolated trusted environment with various security micro-services for developers to build and deploy their trusted applications. TEP environment abstracts various HW Security capabilities making it easy for developer consumption.

1.2 About the Audience

This Design Guide is intended for the SW developers and validation engineers. It captures the architectural description of all the components required to create TEP.

2.0 Intel® TEP Overview

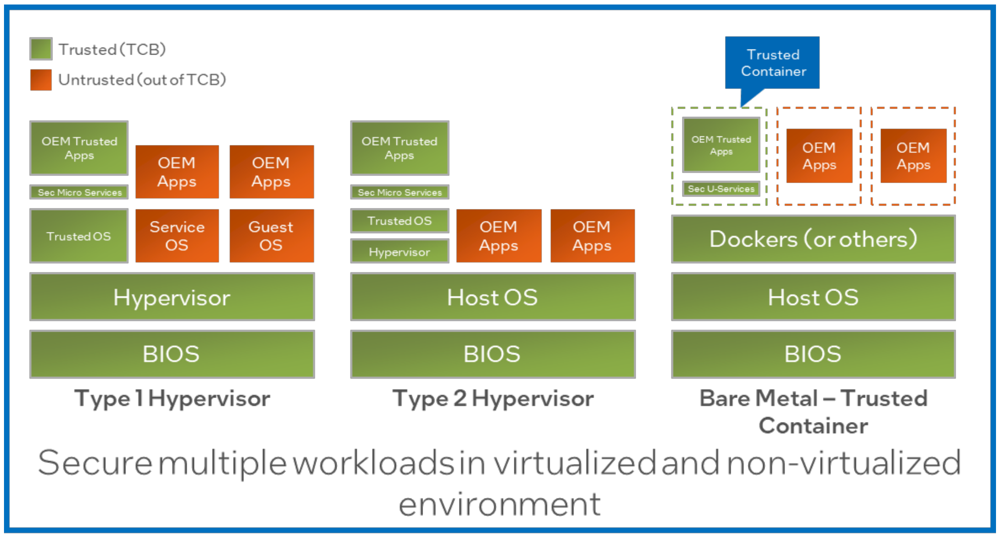
TEP - A framework to speed up, simplify, and reduce the cost of building security into an IoT device from the ground up to demystify implementation choices and bring coherence to the IoT ecosystem. TEP framework is designed to help application developers protect select code and data from disclosure and/or modification.

TEP provides the following support:

- Isolate execution environment for IP workload protection. Supports both virtualized and non-virtualized systems.
- Platform agnostic security micro-services. To simplify security implementation providing customers a trusted application environment.
- Standard API for communication from untrusted to Trusted Environment for easy adoption. Compliant to PKCS#11 API industry.
- Configurable to meet different robustness levels. Provides add-on platform capabilities to meet different robustness levels for different use cases.

TEP provides privileged access, right-size permissions and consistently enforce least-privilege principles with the aim of reducing the complexity, cost, and risk created by stitching together point solutions.

Figure 1 TEP Profiles - Secure multiple workloads in virtualized and non-virtualized environment



TEP Compute Environment provides the following advantages:

- Untrusted applications can interface with Trusted Environment over standard API.

- Trusted applications run in full-fledged Linux kernel based Trusted OS
- No need to re-write the native app when moving to Trusted environment
- Common Micro Services support
- Unified build environment with configurable microservices
- Minimalistic Trusted OS that is hardened
- Deploy Standard Containers (for example, Docker) for native boot environments

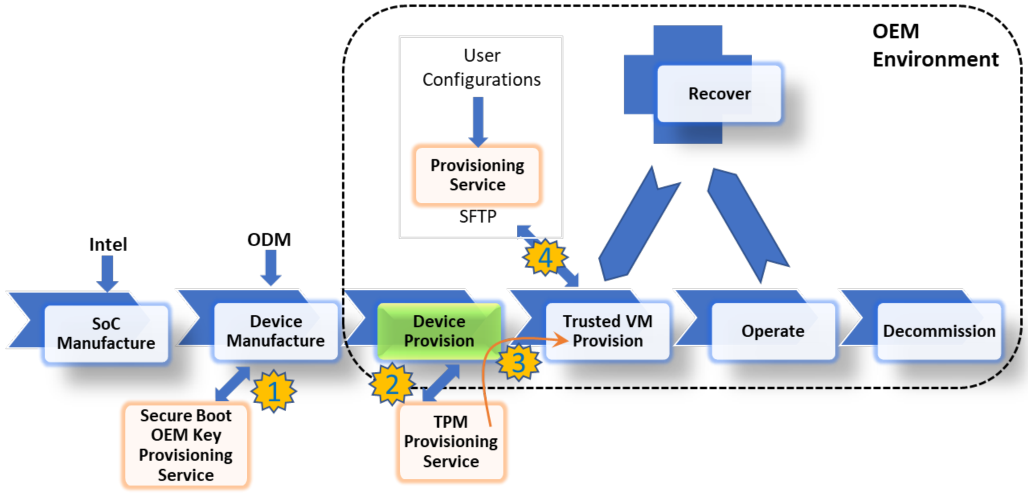
TEP framework supports the provisioning life cycle to ensure that the customers can securely provision their secrets during the early stages of product development to serve as root-of-trust and subsequently used it as the trust base for all its services.

2.1 TEP Provisioning Life Cycle

Figure 2 depicts a representative provisioning life cycle for TEP based system. The provisioning stages are suggestive and can be modified based on the supply chain for each product. It consists of three main provisioning stages:

1. Provisioning by ODM (Original Device Manufacturing) – Optional
2. Provisioning by System Integrator or OEM
3. Provisioning by OEM or End Customer

Figure 2 TEP Provisioning Life Cycle



2.1.1 Provisioning by Original Device Manufacturing (ODM)

Root of trust for Secure boot is provisioned at this stage.

- TEP recommends the platform to enable Intel® Boot Guard, which extends the root-of-trust from Intel Silicon to IBB (Initial Boot Block), the very first stage of BIOS code. Boot Guard requires customers to provision their Public Key Hash of a PKI into Intel Silicon Fuses to serve as root-of-trust for the Secure boot chain.
- TEP also recommends enabling the UEFI Secure boot in the BIOS, which extends the root-of-trust from the second stage of the BIOS all the way to the hypervisor or OS kernel. UEFI secure boot requires to provision the public key of a PKI into UEFI database.

2.1.2 Provisioning by System Integrator (SI) or OEM

OEM ownership credentials to serve as Root of trust for Trusted VM (or Trusted Container) updates is provisioned at this stage.

- SI or OEM to Provision the OEM ownership credentials into the platform TPM at this stage. A ***Symmetric encryption Key (AES-256) for confidentiality and ECC Pub Key (ECC-384)* for integrity** are provisioned into the TPM at this stage. This serves as a root of trust for any Trusted VM or Trusted Container updates.

2.1.3 Provisioning by OEM or End Customer

At this stage the Trusted VM (or Trusted Container) stack is booted up and a one-time provisioning of user configurations and policies are done. Beyond this stage, the Trusted environment is operational. System Admin must encrypt and sign all user configurations using the ownership credentials, which are already provisioned in to the TPM (***Symmetric encryption Key (AES-256) and ECC Public Key (ECC-384)***).

Refer to the *TEP User Guide* for the detailed steps for creation and provisioning of these credentials.

Following are the steps for provision user configurations:

1. Using the *"Secure Config Update"* micro-service, transfer the encrypted User Configuration blob into unencrypted partition of hard disk. Following are few examples of configurations:
   – Signed (ECC Prv Key) PCR-policy
   – Encrypted & Signed User-Config (Network, SSH configurations)
   – Remote Attestation Configurations
   – Any OEM configurations needed for OEM applications
2. Reboot the system.
3. New User config is applied by the Initialization micro-service.
4. Using the **disk encryption** micro-service, the Storage Disk is encrypted with an auto-generated passphrase.
5. **Initialization** micro-service generates a new SSH key pair and stores the private key in encrypted partition of the storage.

3.0 TEP Isolated and Trusted Environment

TEP design consists of Trusted OS, Security Micro Services, and IPC service. TEP also provides an interface library to communicate between VMs (from untrusted VMs/applications to Trusted VM). While Trusted OS and Micro services form the Trusted VM, IPC service takes care of communicating with the Interface Library from the Service VM/Guest VM.

For a Bare Metal profile, the same Security Micro Services run inside of a Trusted Container.

3.1 TEP Architecture

TEP is designed to provide isolated & Trusted environment at HW level using Intel silicon technologies and hypervisor protections to create a higher privileged Trusted Environment. The Trusted VM or the container is launched during the system boot. A lightweight trusted OS runs in the trusted VM to keep the surface attack to minimum. Administrators could host their services within Trusted VM to keep their data confidential and minimize their exposure to threats from other workloads running on the same system.

For a Bare Metal, the trusted environment is a hardened Trusted Container.

Trusted VM (or Trusted Container) offers:

- **Integrity protection** for Trusted OS & its services during boot-up time
- **Run-time protection** for Trusted OS & its services
- **Data at rest protection**
- **Remote Attestation** to verify the integrity of Trusted OS & its services during run-time
- **Secure Configuration Update service**
- **Exclusive access to the TPM for hardware tethering**
- **Unified logging**

Figure 3 TEP Stack for the Virtualized Profile

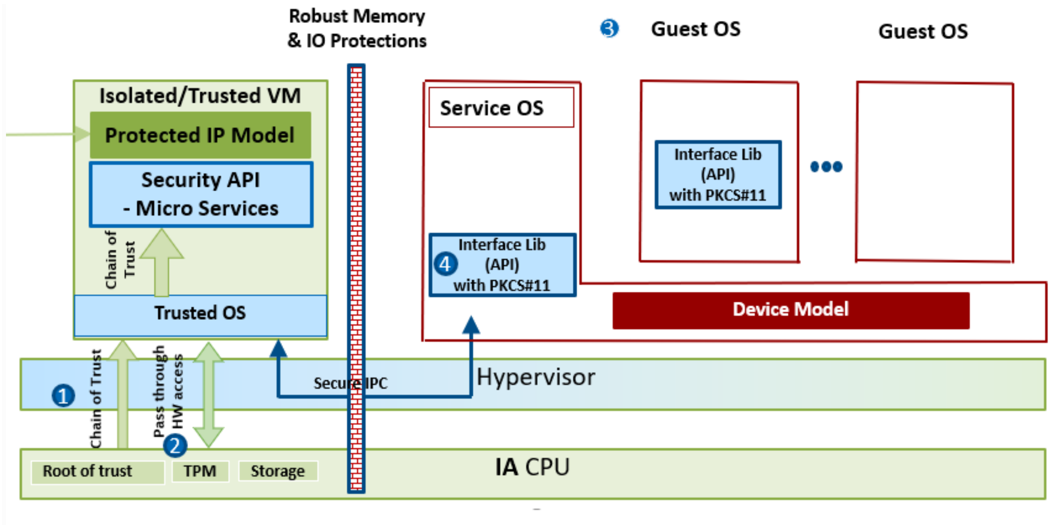
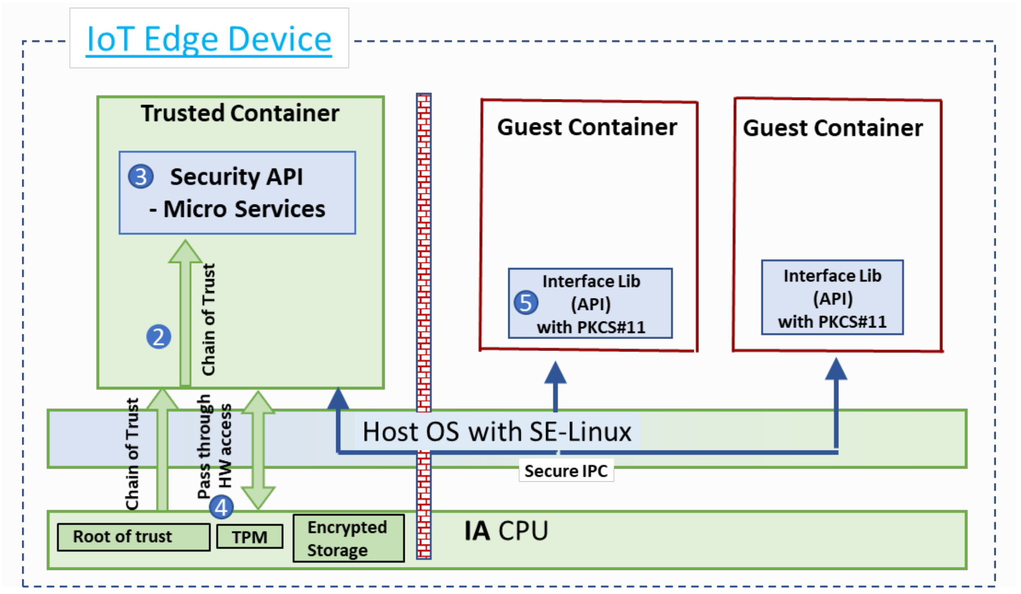


Figure 4 TEP Stack for the Bare Metal with Containers Profile



Some of the components Trusted VM (or Trusted Container) utilizes:

- **Secure boot:** The hypervisor, trusted OS and its file system are verified using secure boot chain of trust, which is rooted to the HW root of trust for boot time integrity protection. The chain of trusted is extended all the way from Hardware root of trust into the application space to ensure the system is always running a known good platform software stack. Following technologies are enabled together to enable the complete chain of trust -- Intel® Boot Guard, UEFI Secure boot, Grub loader Secure boot, Linux DM-verity.

- **Access Controls:** in Bare Metal configuration, the Host OS is enabled with SE-Linux (or AppArmor) as a MAC (Mandatory Access Control) to provide higher isolations between trusted container and rest of the guest containers.
- **Pass-through access to HW:** Trusted VM also has a pass-through access to the PCI devices on the platform providing it exclusive access as per user configuration.
  - **Storage:** Trusted VM is configured to have a direct pass-through access to NVMe based storage device that could be used for exclusive storage. System admin can modify and configure to have other type of storage device (USB, SATA, NVMe, and so on) in the Trusted VM. All other VMs, including Service VM, do not have access to this storage.
  - **TPM:** Trusted VM also has exclusive MMIO pass-through access to the Intel® PTT (Platform trust technology) or discrete TPM on the platform. During the boot up, Trusted VM has the single ownership of the TPM and has a single Storage Root Key and password, one Endorsement Key, and a single set of Platform Configuration Registers (PCRs). Please refer to the section 3.2 "TPM authorization and Keys" for details.
  - **Core Pinning:** Trusted VM is configured to be pinned to specific execution CPU cores. This offers execution level isolation with dedicated L1/L2 caches to applications running in trusted VM.
  - **Network Isolation:** Trusted VM has exclusive access to dedicated ethernet controller, enabling it to have a physically isolated network connection. In a Bare Metal configuration, VLAN service is configured to create a subnet to further isolate the TEP container from rest of the workloads.
  - **IO Isolation:** Trusted VM can have exclusive PCIe pass through to any PCIe devices such as USB, enabling it to have isolated access to sensors & peripherals.

- **Read-Only Mode:** Trusted VM is launched in a Read-Only mode using Ramfs file system. Any changes made to the Trusted VM file system do not persist to storage and always comes back to known-good-state upon a system reboot.

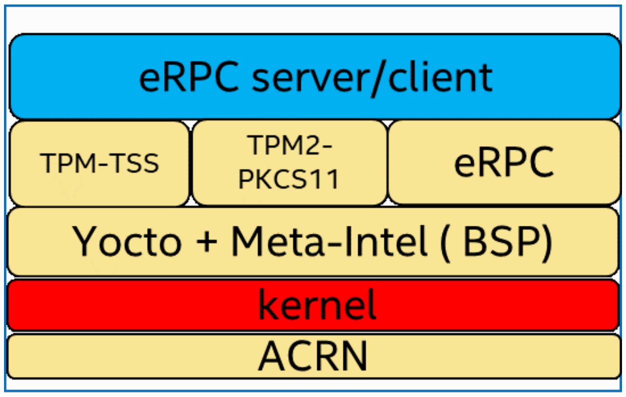
- **Inter VM IPC Communication:** Guest VMs and Service VM can communicate with the Trusted VM and its services using an interface library with PKCS#11 API interface. Standard PKCS#11 API (2.4 version of spec) interfaces are supported. Any OEM application that needs key store services can compile with this interface library and start making PKCS#11 API calls. For Bare Metal configuration, a network-based IPC using Google's GRPC framework is provided for this communication, with PKCS#11 API shim layer for application access. Sample applications are provided to serve as an example how to use these APIs.

3.1.1 Trusted OS

This section describes Trusted OS and a high-level design of its components. Trusted OS is built using Yocto infrastructure but designed to be a lightweight Trusted OS to keep the surface attack minimal. Security micro services that run on Trusted OS includes the TPM TSS stack, TPM2-PKCS#11 stack, eRPC/GRPC stack, LUKS, Initialization scripts and several others. Figure 3 describes different layers of the Yocto recipe which combines to form the Trusted VM image.

Trusted OS and Services are designed to be less than 100MB of image size on disk to keep the attack surface to a minimum.

Figure 5 Trusted OS Component Level View



3.2 TPM Authorizations and Keys

TPM provides following hierarchy authorizations:

1. Platform Authorization:

   The TPM provides the Platform Manufacturer the opportunity to control features it offers to the owner of the TPM. It provides platform hierarchy authorization value (platformAuth) for the Platform manufacturer to control the allocation of NV Indexes, PCR configuration, and availability of the storage, platform, and endorsement hierarchies, changing of the primary seeds, resetting the platformAuth as well as other functions.

2. Owner Authorization:

   The TPM provides the owner authorization value (ownerAuth) as access controls for a platform administrator of a platform with the TPM to manage control of the availability of the storage hierarchies, creating primary keys in the storage hierarchy, NV Indexes, storage, and eviction of persistence of keys, and changing of owner authorizations. Default value of ownerAuth is the Empty Buffer.

3. Endorsement Authorization:

   Endorsement authorization will help prevent unauthorized control of the endorsement hierarchy and unauthorized creation of endorsement primary keys. Remote attestation Identity key (AIK) and attestation certificates are derived from EK.

4. Lockout Authorization:

   Lockout authorization will help to manage the Dictionary Attack parameters and Lockout state. It is also used to reset the storage hierarchy, wiping out all persistent, non-persistent, and permanent keys in that hierarchy. This also erases all NV Indexes associated with the hierarchy and removes their associated index information. TPM2_CLEAR used to clear all hierarchy authorizations and erase all NV Indexes data defaults to lock out authorization.

Note: Currently TEP sets owner, endorsement, and lockout authorizations for TPM. TEP uses same authorization/password (20B random value derived from tpm2_getrandom) to set owner, endorsement, and lockout authorizations to meet Remote Attestation requirement.

3.2.1 SRK and Root Key

TEP creates the following two keys under owner hierarchy:

- Storage Root Key (SRK):

SRK is a primary context with NULL authorization, which is stored at TCG defined shared index "0x81000001" and to be used by all guest/host applications as a primary context to derive keys beneath it. To create and evict SRK at "0x81000001", we need TPM owner authorization.

- Root key:

Root key is primary context to be used only by TEP applications to derive keys beneath it. This is set with 32B random authorization value. And to derive primary context that we need TPM owner authorization.

Note: Refer to Appendix A for list of what all tpm2_tools/commands would and would not need hierarchy authorizations, and what tpm2_tools/commands can be performed using SRK.

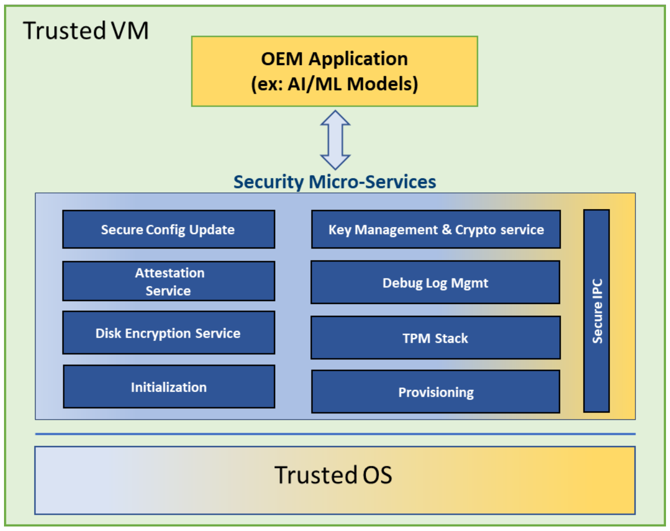
3.2.2 Storage of TPM Authorizations

Before LUKS encrypted drive is setup TPM authorizations are stored in a temporary location at /home/root/tmp/tpm, and after LUKS encrypted drive is setup TPM authorizations are moved to it, whose path is /home/root/tep_luks_dev.

3.3 Security Micro Services

Trusted VM consists of platform agnostic security micro-services, which are implemented with best-known security principles. The intent is to simplify security implementation and provide a ready to use services for customers.

Figure 6 Security Micro Services Component View



3.3.1 Initialization Service

This service is executed during each VM boot and runs the following

1. Checks for configuration updates. If an update is found, it validates the "User Configuration" blob for its cryptographic signature and then decrypts the blob. Then the user configurations are applied to the system.
2. Invokes essential daemons, such as IPC daemon, TPM daemon.
3. Mounts encrypted disk volume.

3.3.2 Remote Attestation Service

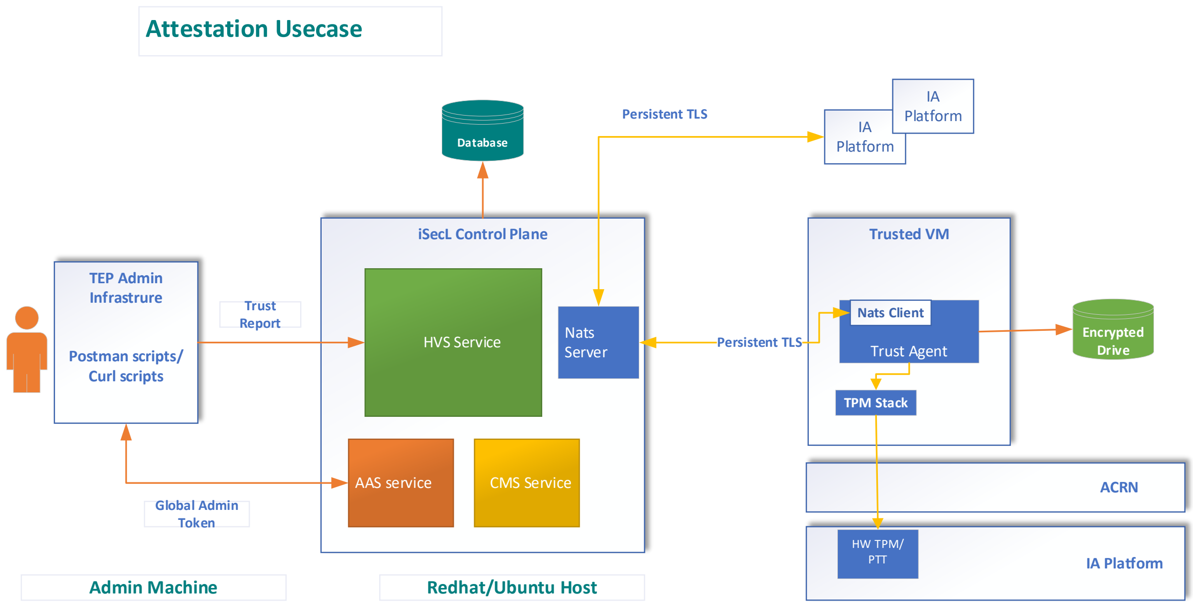
Remote attestation service enables to remotely verify the platform integrity, which includes the platform firmware, BIOS, hypervisor, Trusted OS, security micro-services, and any OEM applications built in to Trusted VM. All the platform software components are cryptographically measured (hashed) and stored inside of TPM PCR (platform config registers), during the platform boot, as per the TCG (Trusted Computing Group) guidance.

TEP uses the iSecL framework for remote attestation service.

Remote Attestation service consists of following components.

- Trustagent component integrated within TEP Trusted VM
- TEP Admin Machine
- Remote Attestation Server (iSecL control plane)

Figure 7 Remote Attestation System View



3.3.2.1 Trust Agent

Trust Agent runs as part of Remote Attestation Service and enables both remote attestation and the extended chain of trust capabilities.

- It collects and provides host specific information
- It provides secure attestation quotes
- Allows secure attestation quotes to be sent to the Verification Service

3.3.2.2 Remote Attestation Server

Remote Attestation Server (also known as iSecL control plane) is the server for attesting the platform integrity. Following are the components required for foundational security:

- Postgres database
- Certificate Management library component
- Authentication and Authorization components
- Host verification service
- NATS Server configuration

It is recommended to follow iSecl github documentation & product guides for building their control plane. Following are the details about the Intel-iSecl.

Git repo: https://github.com/intel-secl/intel-secl

Documentation and Product user Guides:

https://github.com/intel-secl/docs

3.3.2.3 TEP Admin Attestation Infrastructure

TEP Admin have the following tasks for setting up the attestation usecases:

- Get global admin token using the userid and password
- Creation of TEP device flavors,
  - Post flavor group templates - create a flavor template for TEP project and post to HVS.
  - Post flavor group - create a flavor group required and post to HVS.
  - Boot a golden host with Trust agent provisioned. Import flavors from golden host.
- Host registration and generation of reports
  - Register hosts required. Use the host names TA_HOST_ID used in with provisioning Trustagent on TEP devices.
- Following are the various report generation options:
  - Create Trust report - use the TA_HOST_ID. This creates trust report speaking to respective TEP devices
  - List Reports - This generates reports for all available devices registered
  - SAML Report
  - All Hosts - Provides status of all TEP devices connected.
- Creating trustagent.env file required for TEP device Trust agent provisioning.

3.3.3 Disk Encryption Service

Disk encryption service provides data-at-rest protection by using LUKS and dm-crypt setup. This disk encryption service augments the open-source LUKS implementation with platform TPM to harden the solution and provide additional robustness.

Platform Provisioning

1. Initialize disk with LUKS.
2. Create PCR policy.
3. Create signing authority for signing the policies and provision public key in system.
4. Create authorized policy for creating the sealing object.
5. Create secure passphrase and seal to the sealing object.

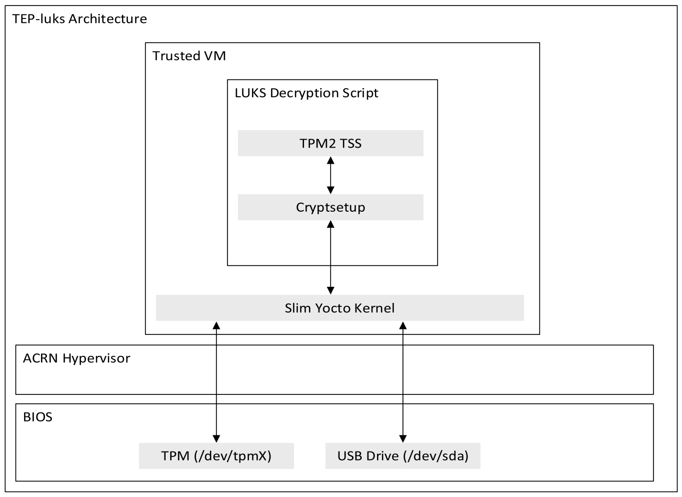
System/ Software integrator

1. Sign valid PCR policies and provide the policy and the signature.

Platform runtime

1. Load signer public key.
2. Verify signature on the PCR policy and generate verification ticket.
3. Satisfy PCR policy and authorized policy.
4. Unseal the secret and pipe to cryptsetup to open the LUKS encrypted volume.

Figure 8 Disk Encryption Service using LUKS and TPM



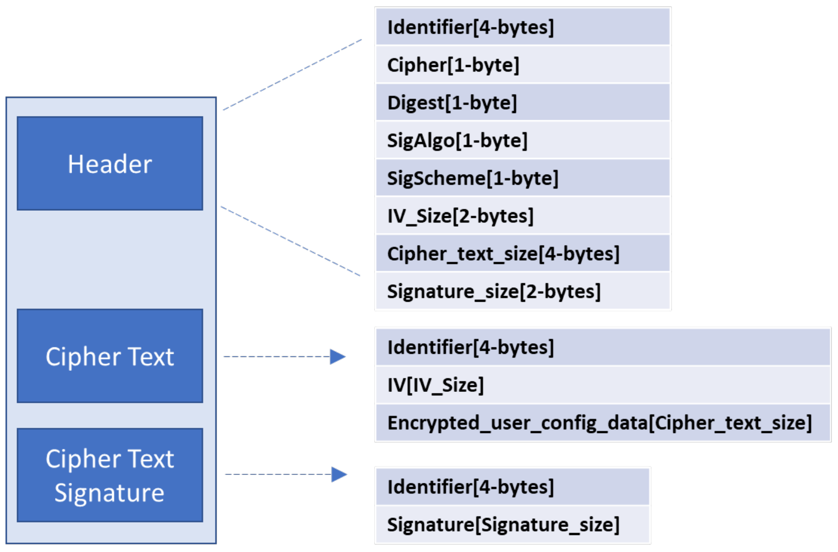
3.3.4 User Config Update Service

User Config update service allows you to update the TEP configurations securely during its operational stage. An example set of configurations that can be included are as follows:

- Signed (ECC Prv Key) PCR-policy
- Encrypted & Signed User-Config (Network, SSH configurations)
- Remote Attestation Configurations
- Any OEM configurations needed for OEM applications

The list of configurations can be extended as per the use case. The data structure for the encrypted and signed configuration blob is as follows:

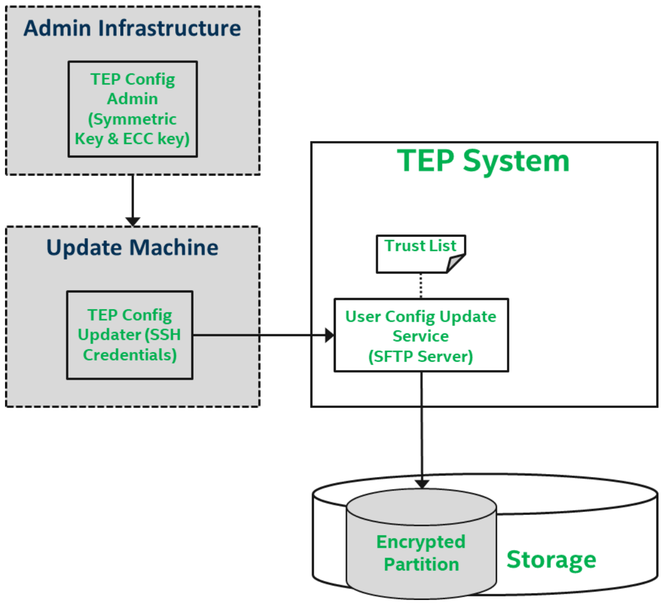
Figure 9 Data Structure of the Signed & Encrypted User Configuration Blob



The system view of this service includes the following:

- **Admin Infrastructure**: Remote admin who has access to ownership credentials (private keys) has the authority to generate the encrypted user config blob.
- **Update Machine**: Is the terminal which has a direct connection to the TEP node and can push the encrypted blob into it using the SSH credentials.
- **TEP System**: Is the target node with TEP enabled.

Figure 10 System View of the User Config Update Infrastructure


Admin Infrastructure
TEP Config Admin (Symmetric Key & ECC key)
Update Machine
TEP Config Updater (SSH Credentials)
TEP System
Trust List
User Config Update Service (SFTP Server)
Encrypted Partition
Storage

3.3.5 Unified Logging

Logging from various services is unified to a common location to ensure confidentiality protection at rest. Debug logs from device provisioning, user config update, disk encryption and TPM2_TSS are logged at /var/log/tep_logs. This allows to further extend this capability to apply user access controls on these logs.

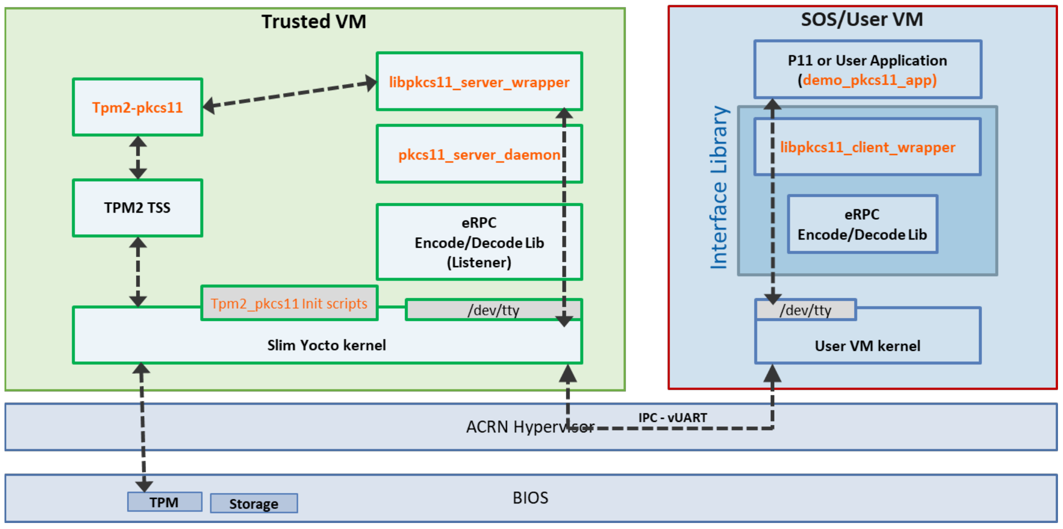
3.3.6 IPC Service with Interface Library

The component level view of the stack describes the Trusted VM components versus the User VM components. A customer application built using the Interface library `/usr/lib/libpkcs11_client_wrapper.so` can make direct calls in to PKCS#11 interface APIs.

For ACRN profile, each application is bound to a virtual UART channel (TTY device) over which it communicates with Trusted VM. The application will have an exclusive lock on the TTY channel and its session until it chooses to release it and at which time a different application can bind itself to that virtual UART channel.

For KVM and BareMetal profile, each application interfaces with Trusted VM (or Container) over a network-based IPC (GRPC – Google's RPC framework). This design supports multiple PKCS11 apps simultaneously talking to Trusted VM or Container.

Figure 11 IPC Service over UART with Interface Library – Component View



The Interface library is an IPC library coupled with a PKCS#11 API shim layer for guest VM applications to communicate with Trusted VM and its crypto services. This library abstracts the details of underlying RPC protocol that implements a client/server mechanism for IPC communication. Guest VM application that build with the interface library have access to the supported set of PKCS#11 APIs for its usage. The publicly available Standard PKCS#11 API (2.4 version of spec).

PKCS #11 (Public-Key Cryptography Standard) defines an application programming interface (API) to cryptographic devices that hold cryptographic information and perform cryptographic functions. These devices are called tokens, and they can be implemented in a hardware or software form. Trusted VM uses the platform TPM as our token.

A PKCS #11 token can store various object types including a certificate; a data object; and a public, private, or secret key. These objects are uniquely identifiable through the PKCS #11 URI scheme.

A PKCS #11 URI is a standard way to identify a specific object in a PKCS #11 module according to the object attributes. This enables you to configure all libraries and applications with the same configuration string in the form of a URI.

Following object types are supported:

- RSA Support.
- AES Support.
- RNG Support.

- Object Management support.
- HMAC Support.

The following table lists the PKCS11 APIs are now supported:

C_Initialize	C_Finalize	C_Getinfo
C_InitToken	C_GetTokenInfo	C_GetSlotList
C_GetMechanismInfo	C_OpenSession	C_Login
C_InitPIN	C_Logout	C_CloseSession
C_GenerateKeyPair	C_GetAttributeValue	C_SignInit
C_Sign	C_VerifyInit	C_Verify
C_GenerateRandom	C_EncryptInit	C_Encrypt
C_EncryptUpdate	C_EncryptFinal	C_DecryptInit
C_Decrypt	C_DecryptUpdate	C_DecryptFinal
C_FindObjects	C_FindObjectsInit	C_FindObjects
C_FindObjectsFinal	C_SignUpdate	C_SignFinal
C_VerifyUpdate	C_VerifyFinal	C_GetFunctionList
C_DestroyObject	C_CreateObject	C_GetSessionInfo
C_SetPIN	C_CloseAllSessions	C_GetMechanismList
C_DigestInit	C_Digest	C_DigestUpdate
C_DigestFinal	C_SeedRandom	

4.0 Intel Recommendations

Following are Intel design recommendations for system security.

- Keep the Trusted OS, GRUB and BIOS stacks up to date with patches.
- In the production system:

1. Close all debug interfaces, including JTAG, Serial connections and Shell access.
2. Disable ACRN VMM dump.
3. BIOS Menu lock down with password.
4. Out-Of-band provisioning of UEFI keys must be disabled.
5. Please change the default passwords for the users (root/ladmin/guest/user).

   For details, refer the *TEP User Guides.*

- Recommendation to use MAC (Mandatory Access Control) controls on Service VM to protect the User PIN for PKCS.
- To protect against an adversary with physical access, enable TME (total memory encryption) the disk encryption.
- *"Users should keep systems up to date with the latest firmware and guard systems against unauthorized physical access. Systems where end of manufacturing was performed by the OEM and where Intel Firmware Version Control technology (hardware anti-rollback) was enabled are at far less risk.*

Appendix A Tool Commands and Their Required Hierarchy Authorization

The following tables describes the authorization of TPM2_TOOLS and Commands.

Table 2: TPM2_TOOLS/Commands which require hierarchy authorization

SNo	Tool/Command	Required hierarchy authorization	Comments
1	tpm2_changeauth	The old hierarchy authorization value specified with **-c**	
2	tpm2_changeeps	Platform	
3	tpm2_changepps	Platform	
4	tpm2_clear	Platform or lockout	By default, it operates on the lockout hierarchy, operating on platform hierarchy require platform authentication
5	tpm2_clearcontrol	Platform	
6	tpm2_clockrateadjust	Owner	
7	tpm2_createak	Endorsement	
8	tpm2_createek	Endorsement	
9	tpm2_createprimary	Owner	
10	tpm2_dictionarylockout	Lockout	
11	tpm2_evictcontrol	Owner	
12	tpm2_getcommandauditdigest	Endorsement	
13	tpm2_hierarchycontrol	Platform	
14	tpm2_pcrallocate	Platform	
15	tpm2_setclock	Owner	
16	tpm2_setcommandauditstatus	Owner	
17	tpm2_setprimarypolicy	Respective hierarchy auth	
18	tpm2_nvdefine	Owner	
19	tpm2_nvcertify	Owner	
20	tpm2_nvextend	Owner	
21	tpm2_nvincrement	Owner	
22	tpm2_nvsetbits	Owner	
23	tpm2_nvread	Owner	
24	tpm2_nvreadlock	Owner	
25	tpm2_nvundefine	Owner	
26	tpm2_nvwrite	Owner	
27	tpm2_nvwritelock	Owner	
28	tpm2_policyauthorizenv	Owner	
29	tpm2_policynv	Owner	

Table 3: TPM2_TOOLS/Commands which does not require owner authorization with SRK

SNo	Tool/Command	Comments
1	tpm2_activatecredential	
2	tpm2_certify	
3	tpm2_certifycreation	
4	tpm2_changeauth	The old authorization value for the TPM object created under SRK specified with -c
5	tpm2_commit	
6	tpm2_create	Can use SRK with NULL authorization as primary context to derive keys
7	tpm2_duplicate	
8	tpm2_ecdhkeygen	
9	tpm2_ecdhzgen	
10	tpm2_encryptdecrypt	
11	tpm2_getsessionauditdigest	
12	tpm2_gettime	
13	tpm2_zgen2phase	
14	tpm2_hmac	
15	tpm2_import	
16	tpm2_load	
17	tpm2_loadexternal	Defaults to NULL hierarchy
18	tpm2_makecredential	
19	tpm2_quote	
20	tpm2_rsadecrypt	
21	tpm2_rsaencrypt	
22	tpm2_sign	

Table 4: TPM2_TOOLS/Commands which does not require hierarchy authorization

SNo	Tool/Command	Comments
1	tpm2_incrementalselftest	
2	tpm2_getrandom	
3	tpm2_getcap	
4	tpm2_certifyX509certutil	
5	tpm2_checkquote	
6	tpm2_createpolicy	

SNo	Tool/Command	Comments
7	tpm2_ecephemeral	
8	tpm2_eventlog	
9	tpm2_flushcontext	
10	tpm2_geteccparameters	
11	tpm2_getekcertificate	
12	tpm2_getpolicydigest	
13	tpm2_gettestresult	
14	tpm2_pcrevent	
15	tpm2_pcrextend	
16	tpm2_pcrread	
17	tpm2_pcrreset	
18	tpm2_policyauthorize	
19	tpm2_nvreadpublic	
20	tpm2_policyauthvalue	
21	tpm2_policycommandcode	
22	tpm2_policycountertimer	
23	tpm2_policycphash	
24	tpm2_policyduplicationselect	
25	tpm2_policylocality	
26	tpm2_policynamehash	
27	tpm2_verifysignature	
28	tpm2_policynvwritten	
29	tpm2_policyor	
30	tpm2_policypassword	
31	tpm2_policypcr	
32	tpm2_policyrestart	
33	tpm2_policysecret	
34	tpm2_policysigned	
35	tpm2_policytemplate	
36	tpm2_policyticket	
37	tpm2_print	
38	tpm2_rc_decode	
39	tpm2_readclock	
40	tpm2_readpublic	
41	tpm2_selftest	

SNo	Tool/Command	Comments
42	tpm2_send	
43	tpm2_sessionconfig	
44	tpm2_shutdown	
45	tpm2_startauthsession	
46	tpm2_startup	
47	tpm2_stirrandom	
48	tpm2_testparms	
49	tpm2_unseal	
50	tpm2_hash	