+ 14 Nov 2024

Islet Project
Application provisioning

Piotr Radosław Sawicki

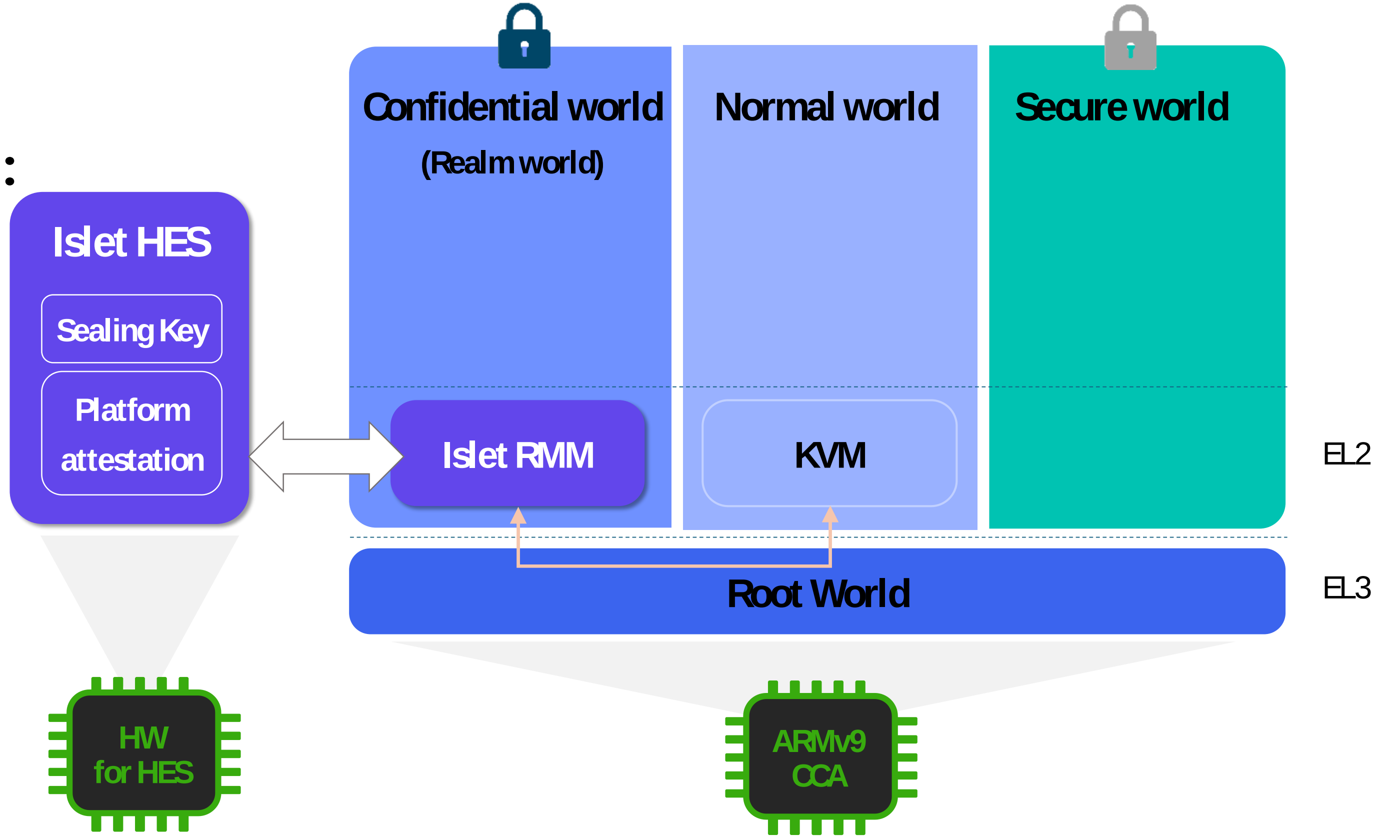
Samsung Research

Agenda

1. Introduction
2. Application provisioning architecture
3. Provisioning details
4. Realm metadata
5. Derivation of sealing keys

A Rust-based CC platform SW on Arm CCA to enable CC on end-user devices

- **Realm Management Monitor (RMM) :** runs confidential VMs, aka realms, in a separate world
- **Hardware Enforced Security (HES) :** provides TCB integrity measurement, platform attestation and keys for data sealing



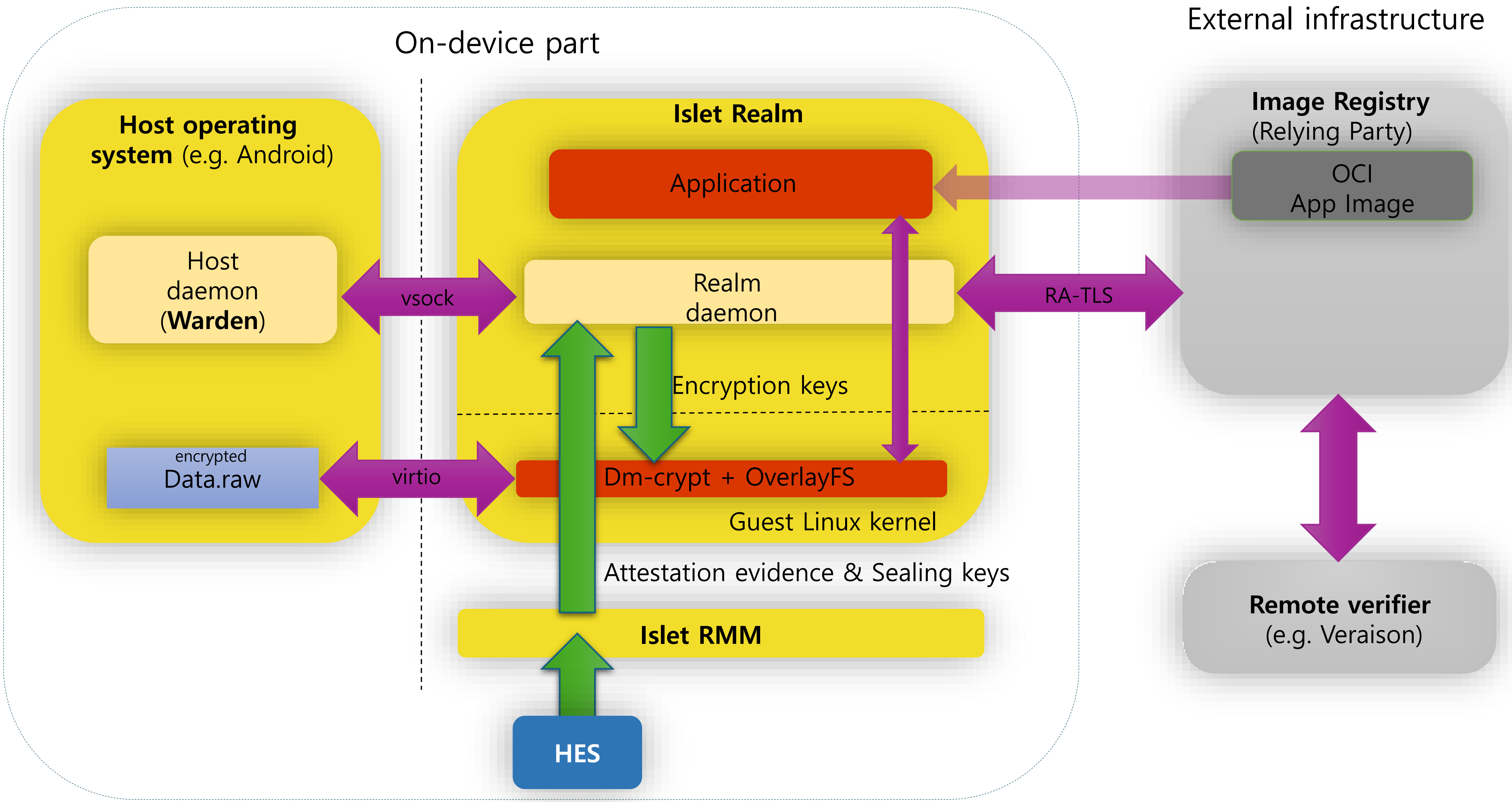
Main goals

- The mechanism should should allow to securely provision applications into Realm VMs from external application registries
- The off-line mode should be supported
- The solution should target Linux-based environments

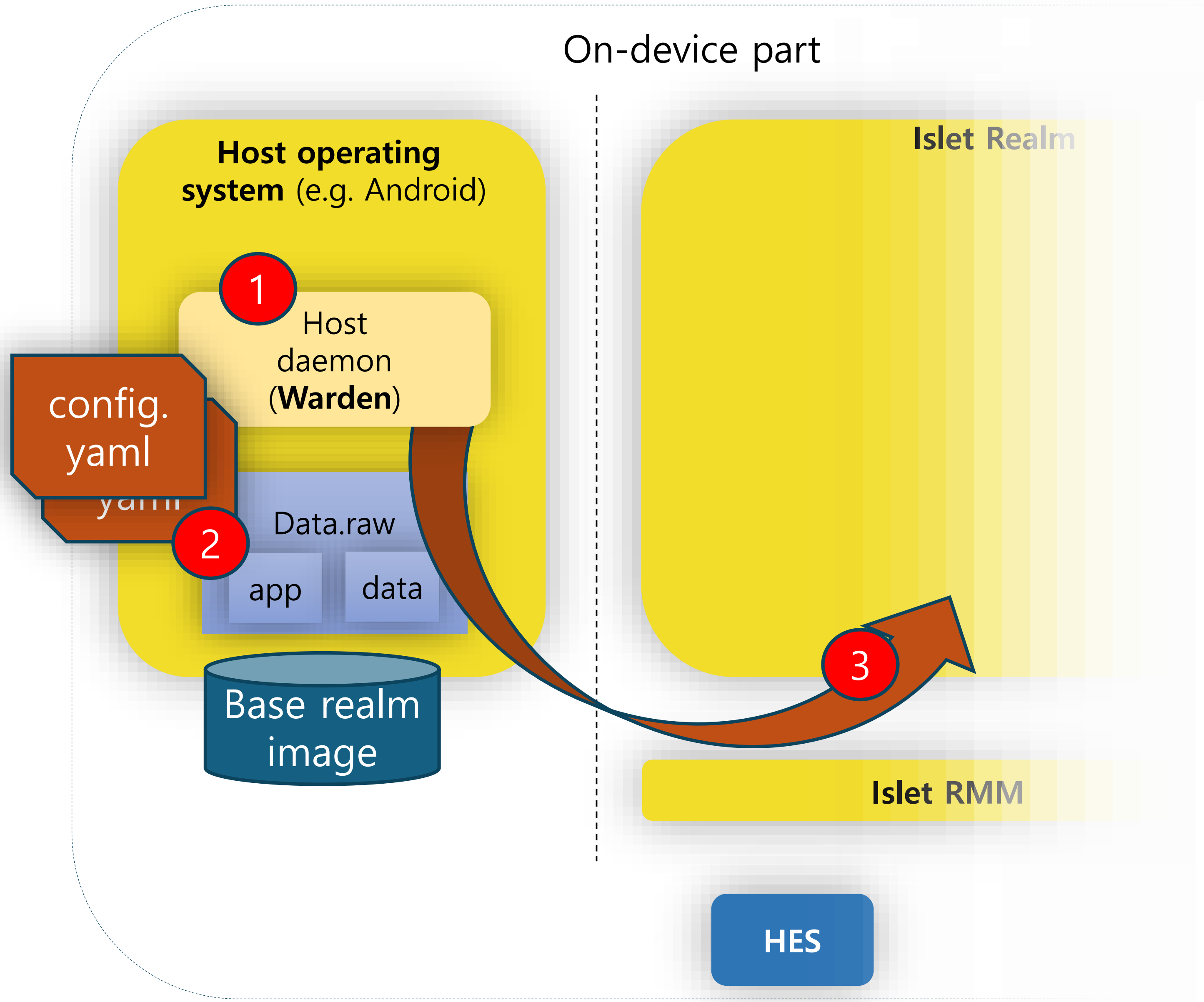
Key ideas

- Split the Realm content into two parts:
  - a lightweight base realm image containing a dedicated service responsible for provisioning
  - applications provisioned from external application registries
- Use the OCI (Open Container Initiative) image format for packaging applications
- The provisioning involves setup of a secure channel using the RA-TLS protocol that combines remote attestation with TLS
- Secure encrypted storage to support off-line mode
- Data encryption keys are sealed to the platform and the realm content

Application provisioning architecture



Provisioning details



External infrastructure

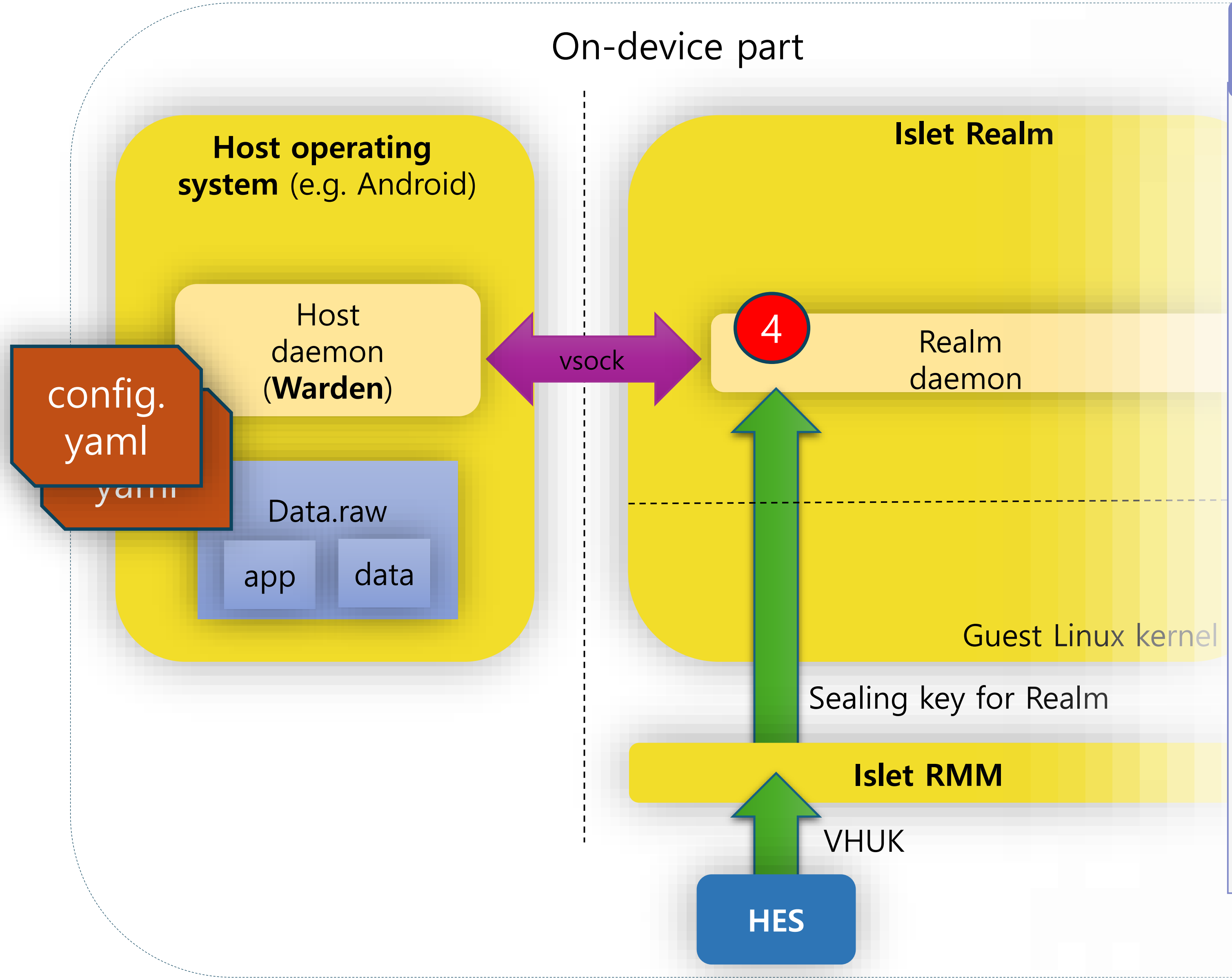
Warden daemon

1. The client connects to the Warden daemon and requests creation of a new realm instance. The client also provides application provisioning information that contains the URL of the image registry, application id, version.

2. Warden prepares an empty application disk image, and saves the configuration data for off-line usage.

3. Warden launches a Realm VM by providing configuration options, the path to the application disk and the base realm image. The base realm image is a lightweght Linux distribution that contains the Realm daemon.

Provisioning details



Realm daemon

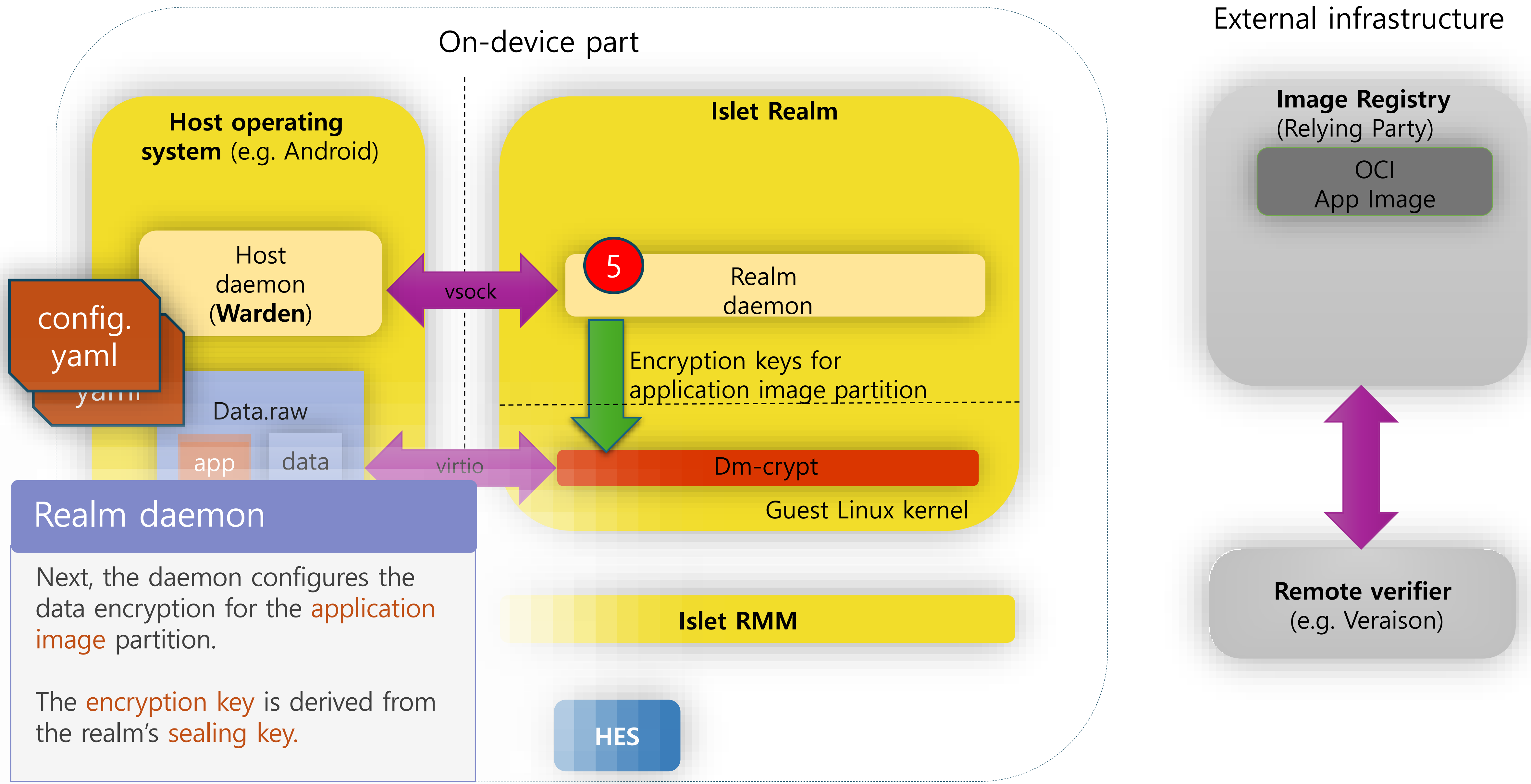
Before the launch of the realm, its content is measured by the Islet RMM to produce the Realm Initial Measurement (RIM) to reflect the initial state of the realm in the the attestation evidence.

The guest Linux OS is started altogether with the Realm daemon. On startup, the daemon reads the sealing key from Islet RMM.

This sealing key will be used in further derivations of data encryption keys for application disk partitions.

Then, the Realm daemon establishes a connection with the Warden demon & gets the application provisioning data.

Provisioning details

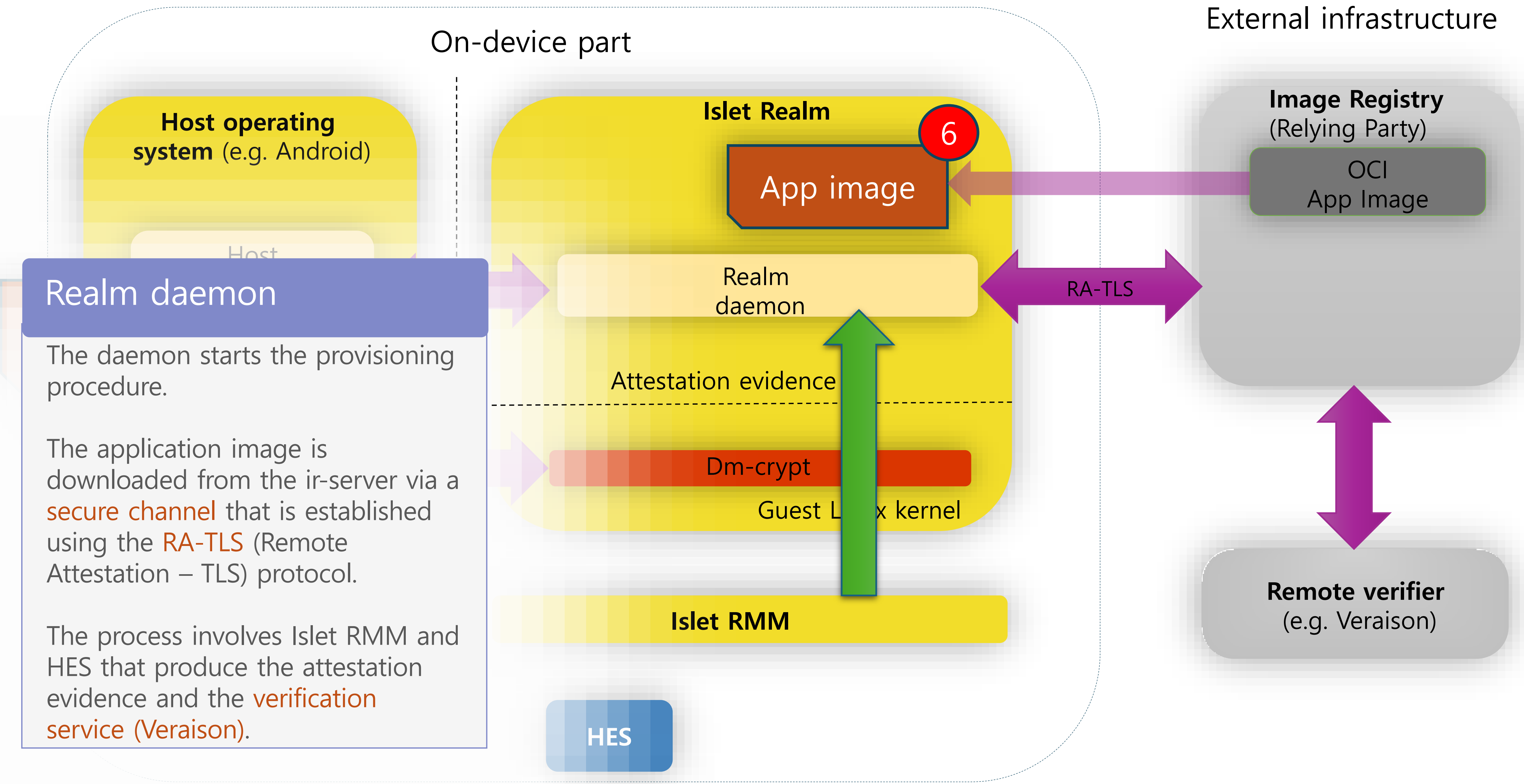


Realm daemon

Next, the daemon configures the data encryption for the application image partition.

The encryption key is derived from the realm's sealing key.

Provisioning details



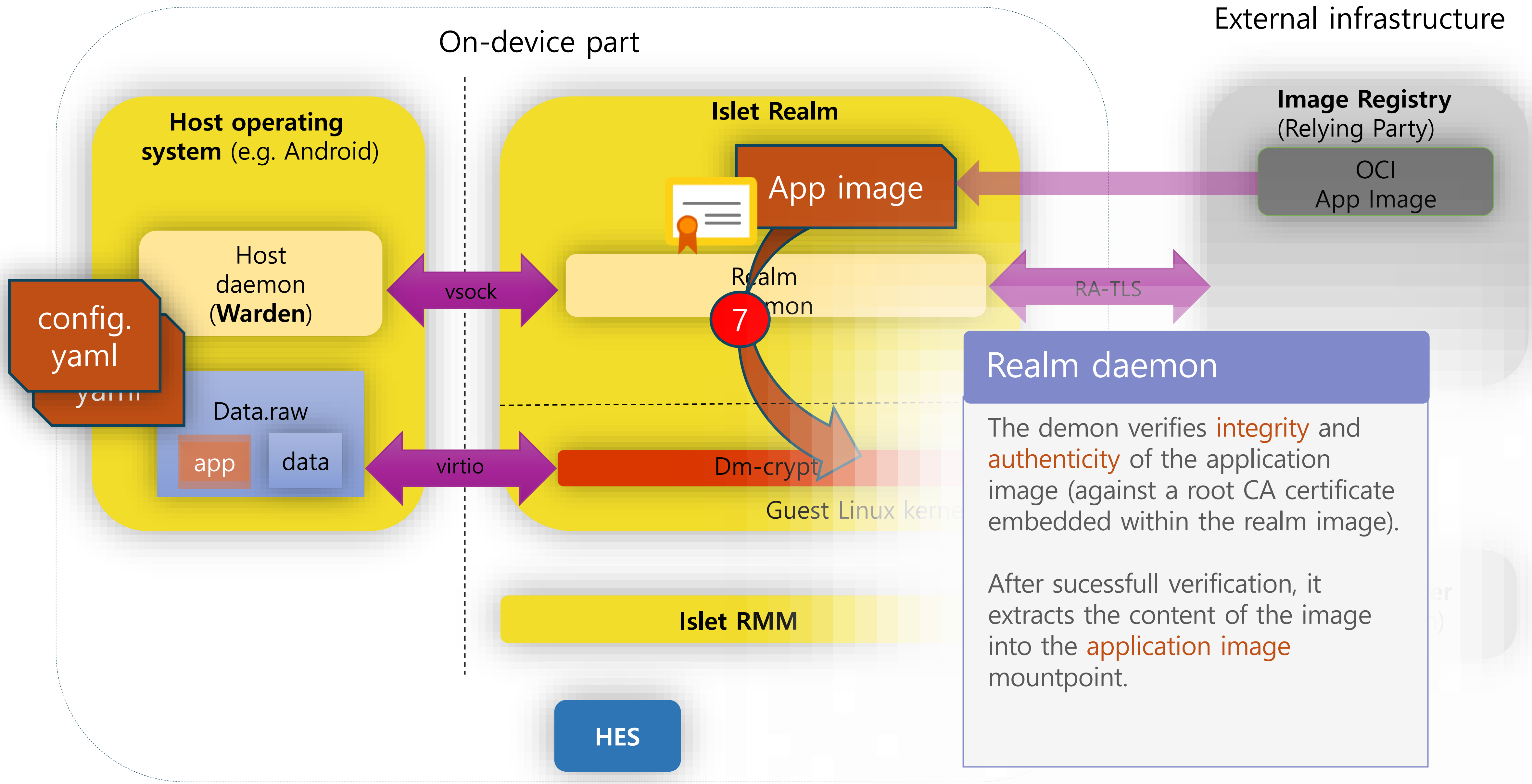
Realm daemon

The daemon starts the provisioning procedure.

The application image is downloaded from the ir-server via a secure channel that is established using the RA-TLS (Remote Attestation – TLS) protocol.

The process involves Islet RMM and HES that produce the attestation evidence and the verification service (Veraison).

Provisioning details

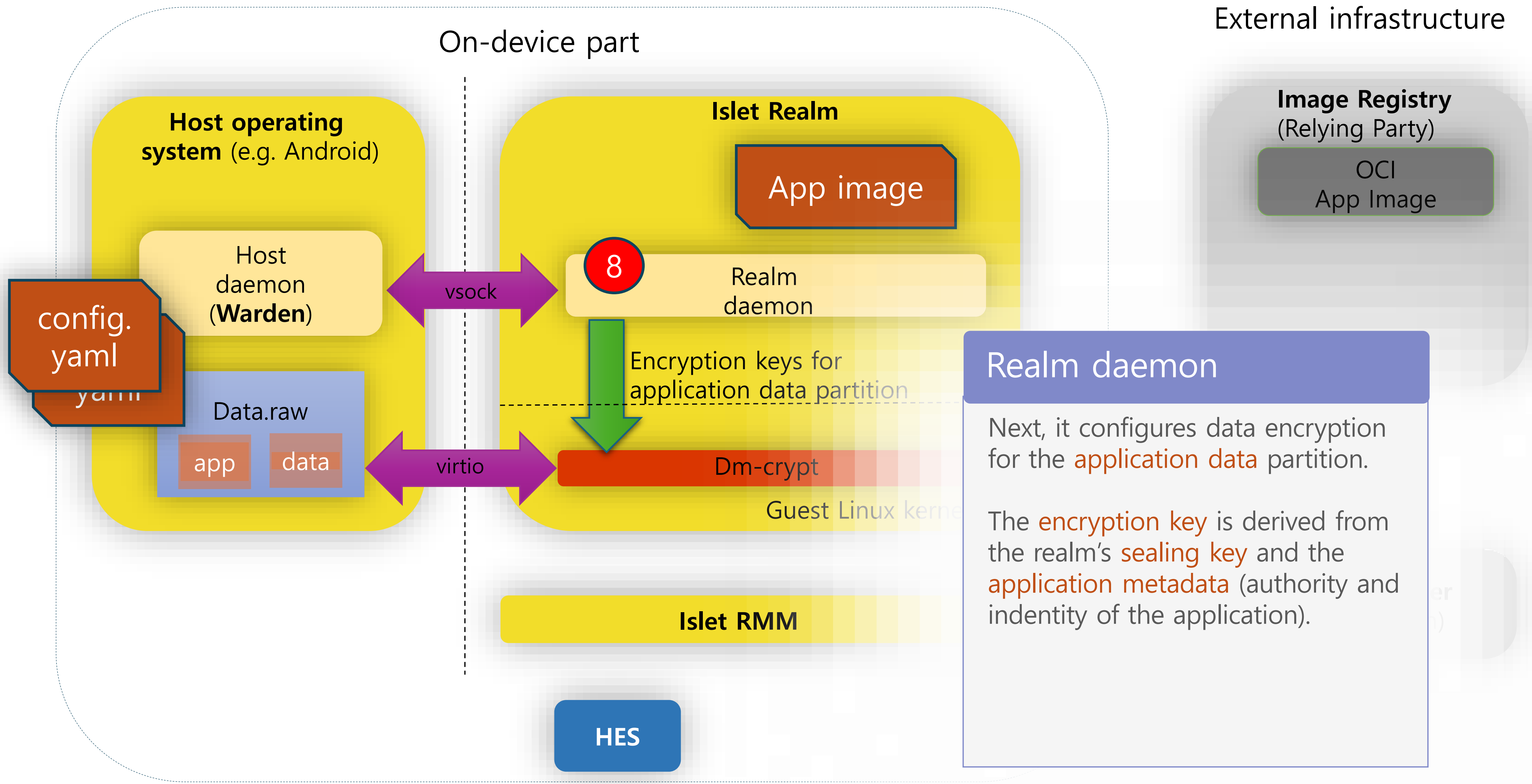


Realm daemon

The demon verifies integrity and authenticity of the application image (against a root CA certificate embedded within the realm image).

After sucessfull verification, it extracts the content of the image into the application image mountpoint.

Provisioning details

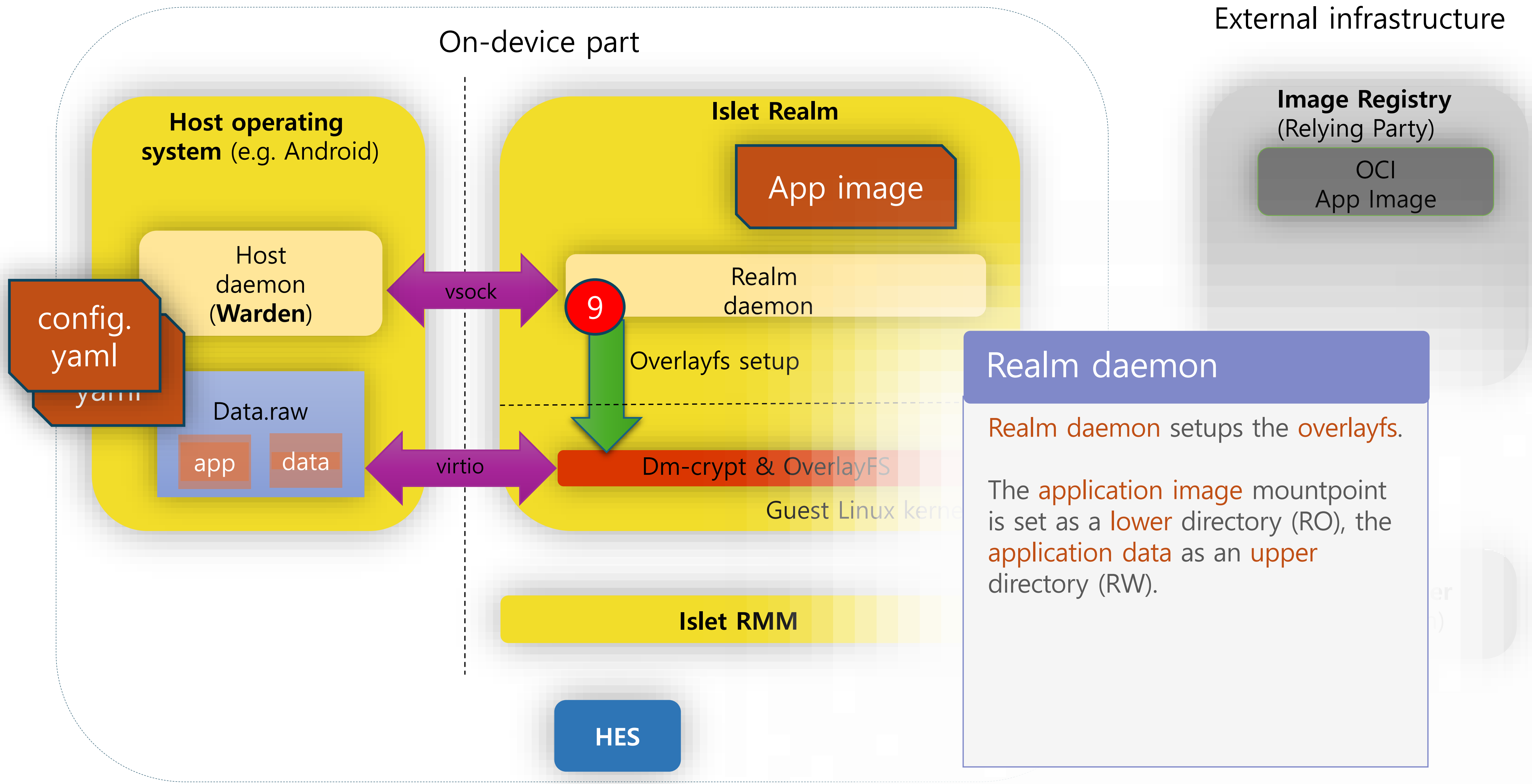


Realm daemon

Next, it configures data encryption for the application data partition.

The encryption key is derived from the realm's sealing key and the application metadata (authority and indentity of the application).

Provisioning details



Realm daemon

Realm daemon setups the overlayfs.

The application image mountpoint is set as a lower directory (RO), the application data as an upper directory (RW).

Provisioning details

On-device part

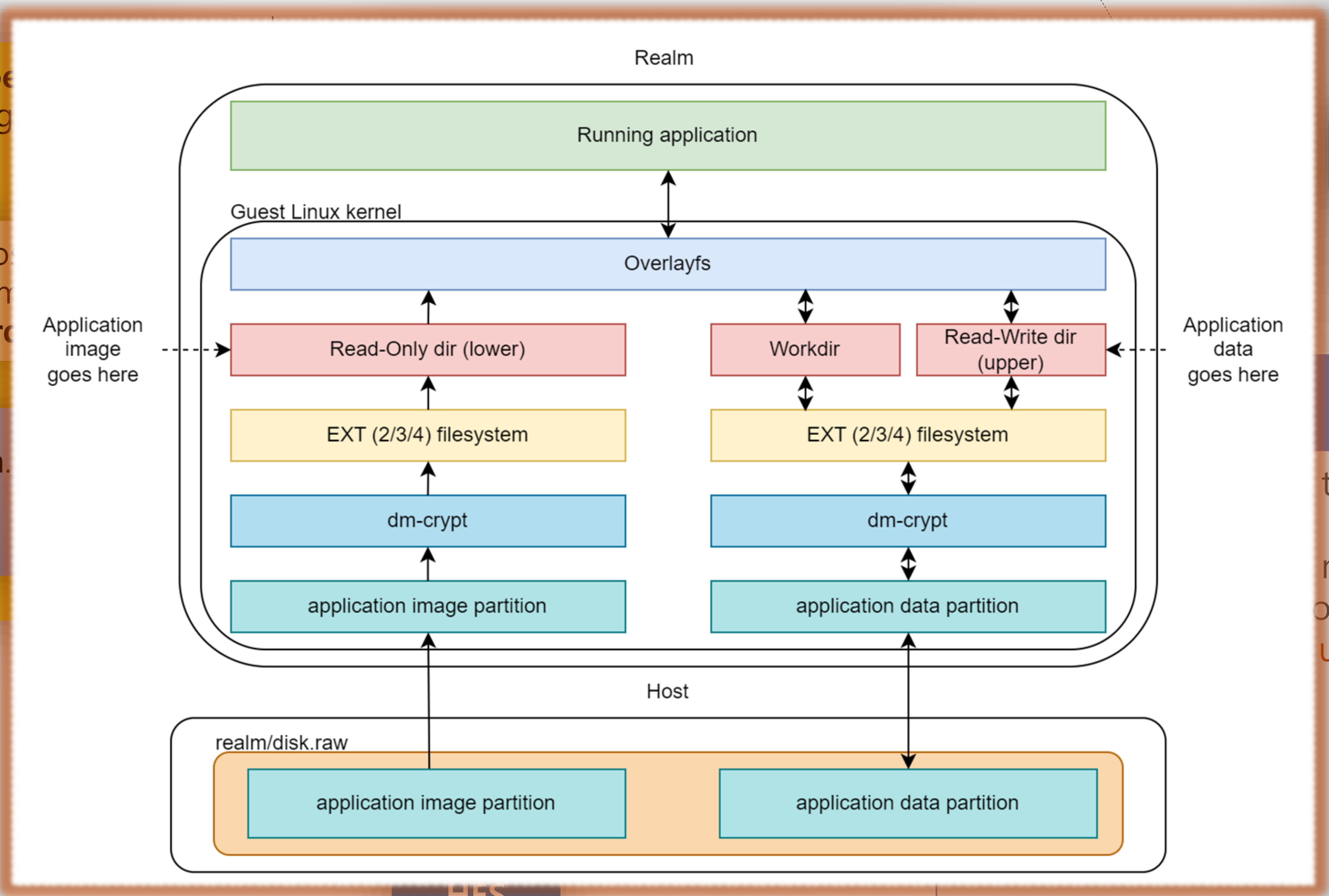
Host ope
system (e.g

Ho
daem
(Warc

config.
yaml

Data.

app



HES

External infrastructure

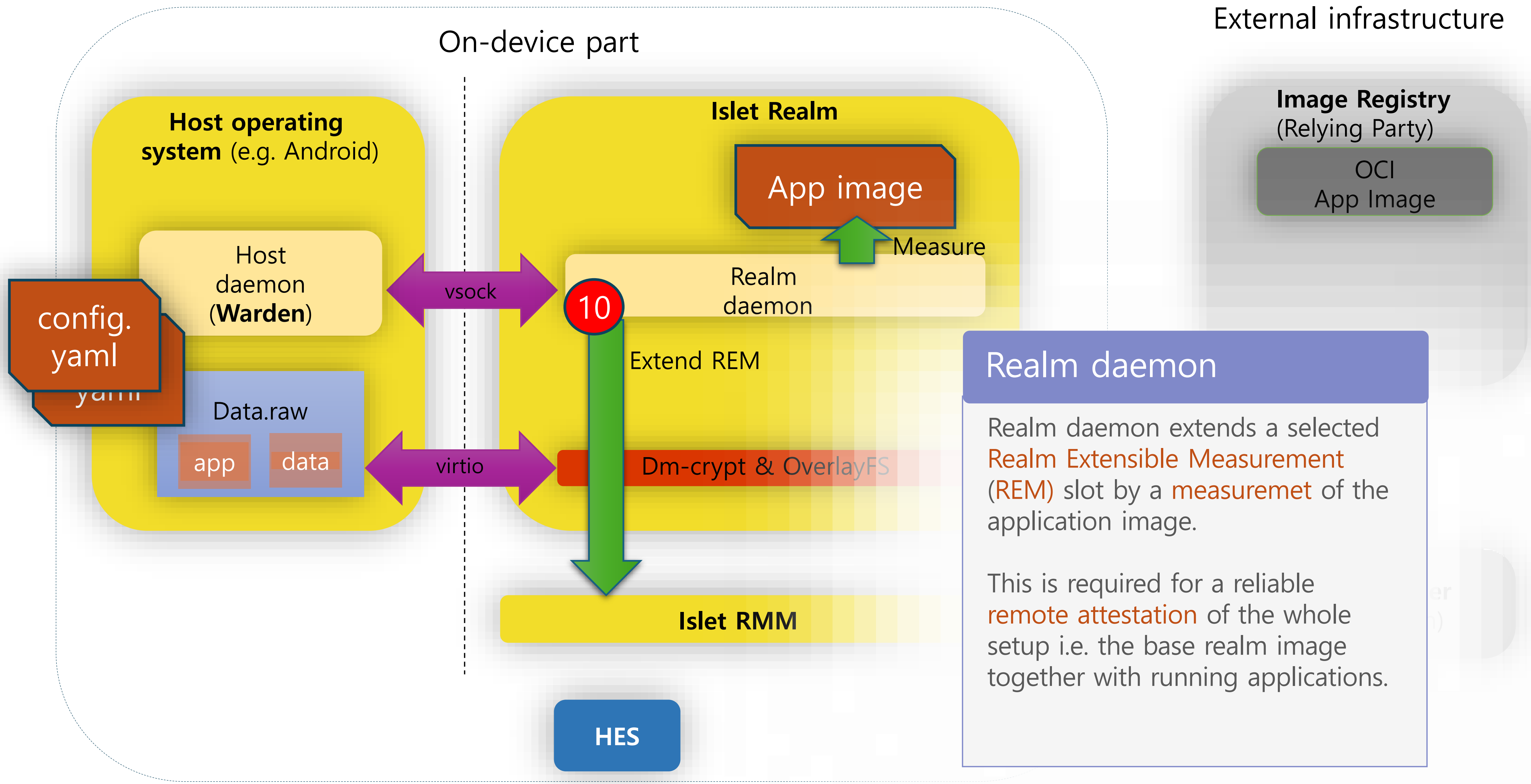
Image Registry
(Relying Party)

OCI
App Image

the overlayfs.

mountpoint
ory (RO), the
upper

Provisioning details



Realm daemon

Realm daemon extends a selected Realm Extensible Measurement (REM) slot by a measuremet of the application image.

This is required for a reliable remote attestation of the whole setup i.e. the base realm image together with running applications.

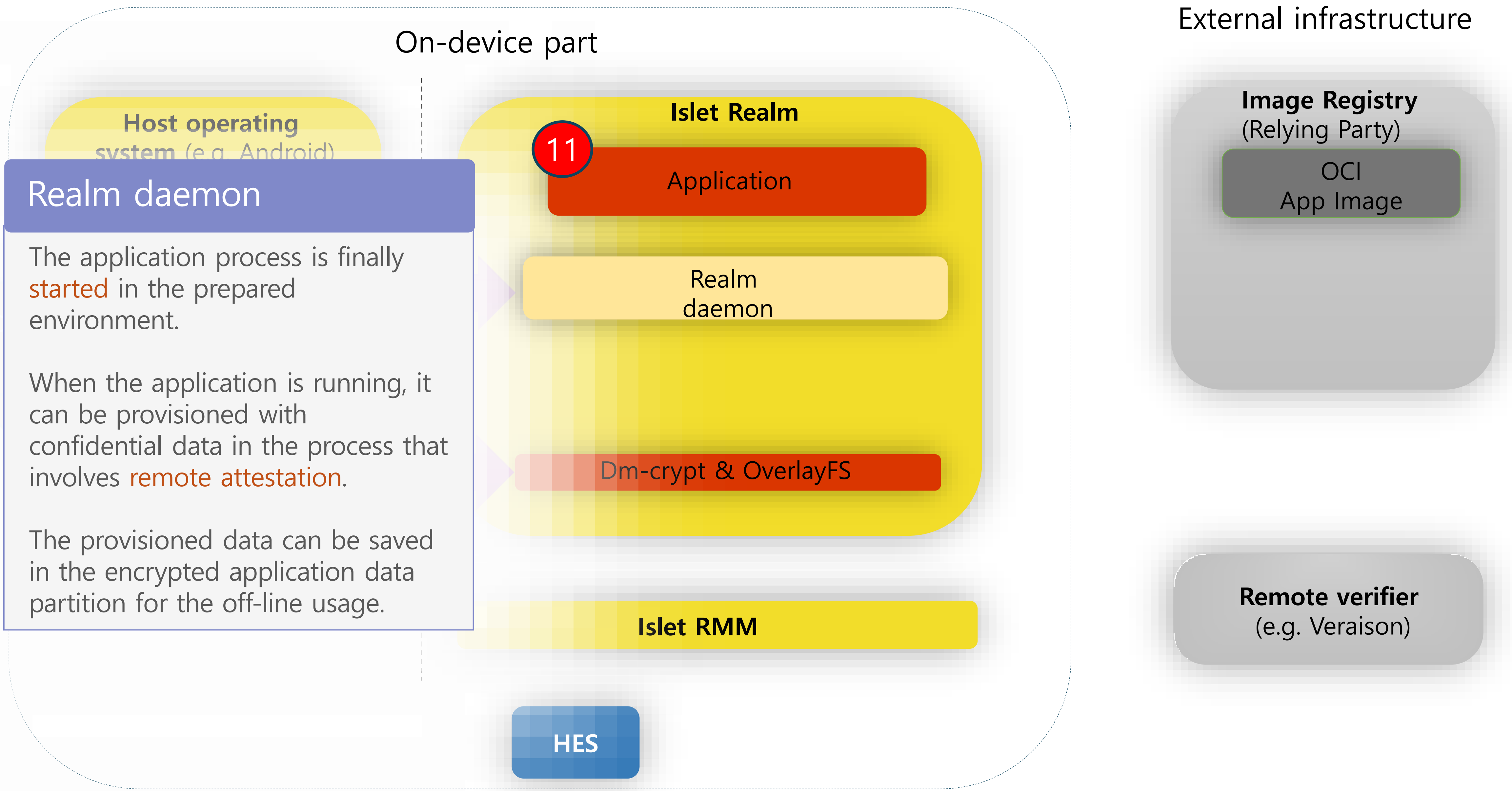
Provisioning details

Realm daemon

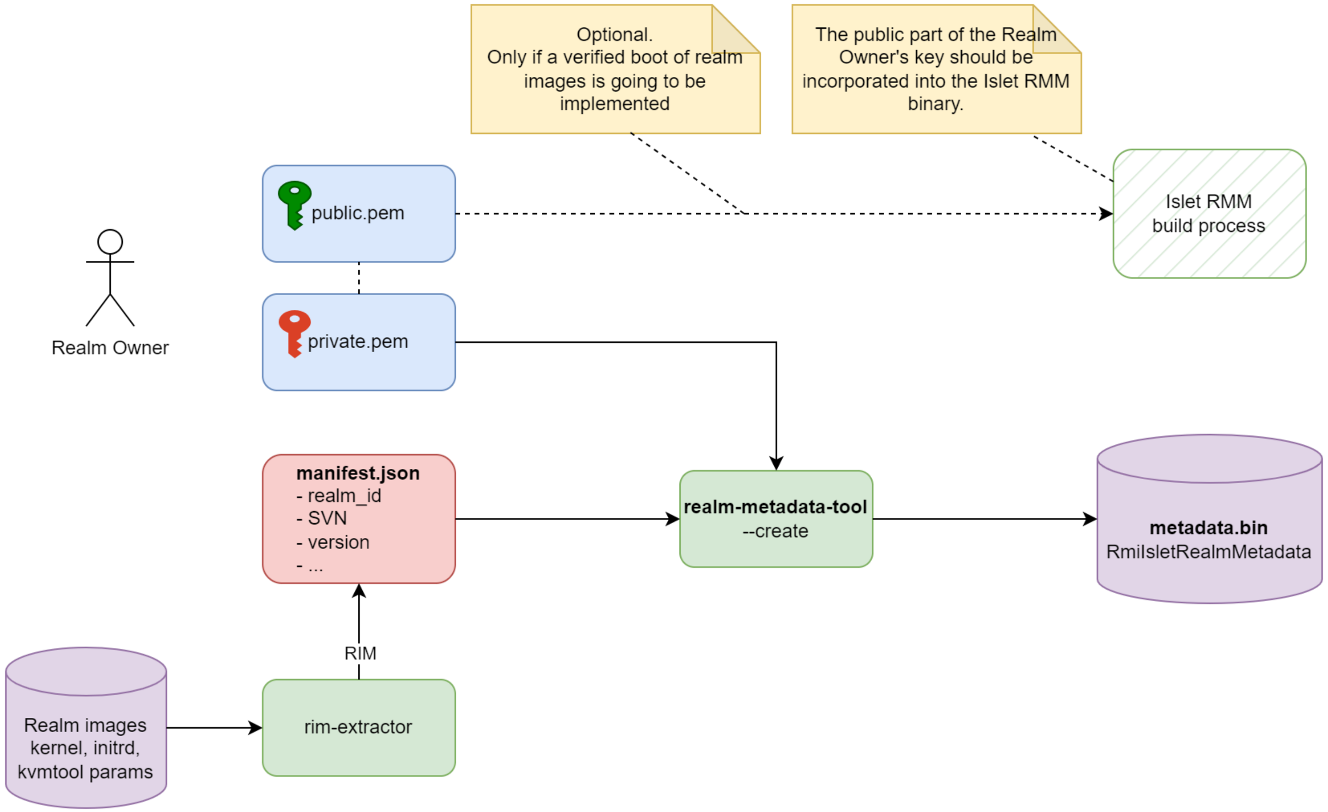
The application process is finally started in the prepared environment.

When the application is running, it can be provisioned with confidential data in the process that involves remote attestation.

The provisioned data can be saved in the encrypted application data partition for the off-line usage.



Realm metadata – needed for stable data sealing



Realm metadata

Realm metadata is a digitally signed binary object that contains following information:

- Realm identifier
- RIM
- Version and Security Version Number
- The public key of the Realm vendor (used to verify the signature)

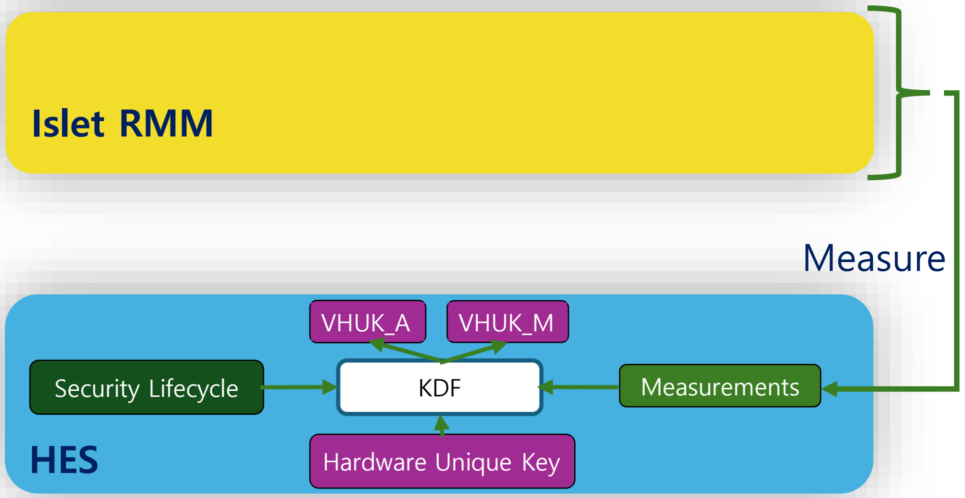
Realm metadata is used in sealing keys derivation process to provide stable keys immune to realm updates.

It can be used to implement a verified boot of the realm image.

The realm metadata file is produced using a dedicated realm metadata tool, and provisioned to the Islet RMM during the startup of a realm (kvmtool, kvm are involved).

Islet RMM is responsible of verifying of the signature and the associated RIM of the metadata block.

Derivation of Virtual Hardware Unique Keys



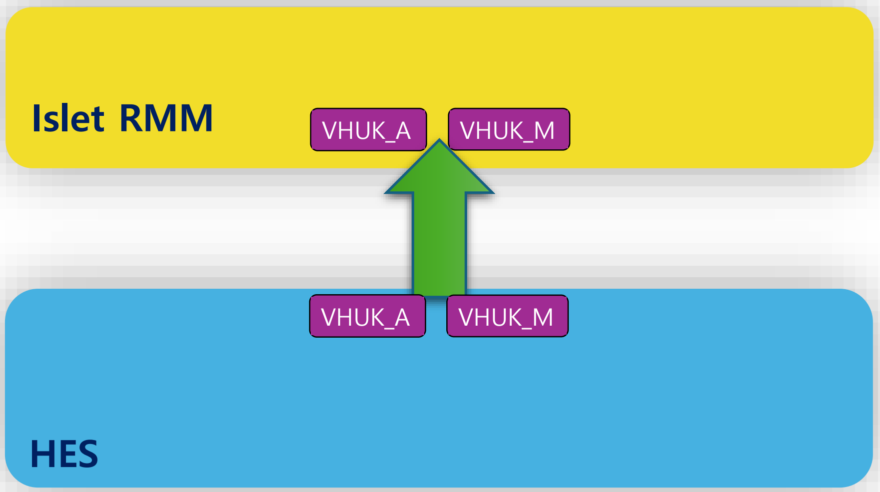
HES

During the platform startup, HES derives two **Virtual Hardware Unique Keys**:

- Authority based VHUK (VHUK_A) – derived from Hardware Unique Key (HUK), Security Lifecycle state and authority data of firmware (signer id and name of measured firmware components)
- Measurement based VHUK (VHUK_M) – derived from HUK, Lifecycle and all measurements of the firmare components

As a Key Derivation Function (KDF) we use a Counter Mode KDF where the pseudo-random function (PRF) is based on SHA-256 and AES-ECB. This KDF is described in NIST SP800-108. It's the same function that is already used in the Realm Attestation Key (RAK) derivation process.

Islet RMM initialization

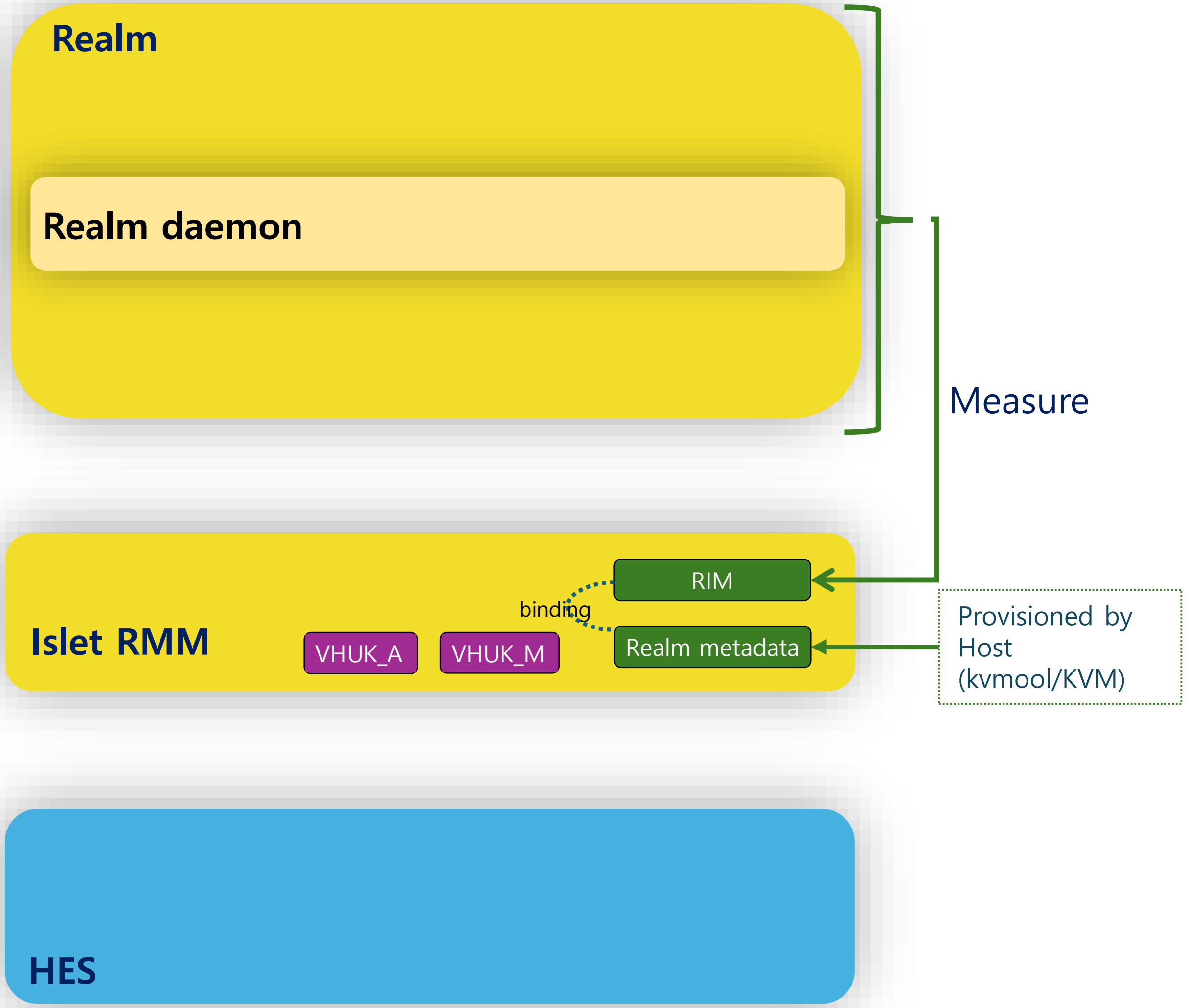


Islet RMM

During the Islet RMM initialization, The VHUK_A and VHUK_M are fetched from HES using a dedicated PSA API (RSS_VHUK_GET_KEY).

RMMD (TF-A) implements additional SMC exposed to Islet RMM for fetching Virtual Hardware Unique Keys.

Initialization of Realm



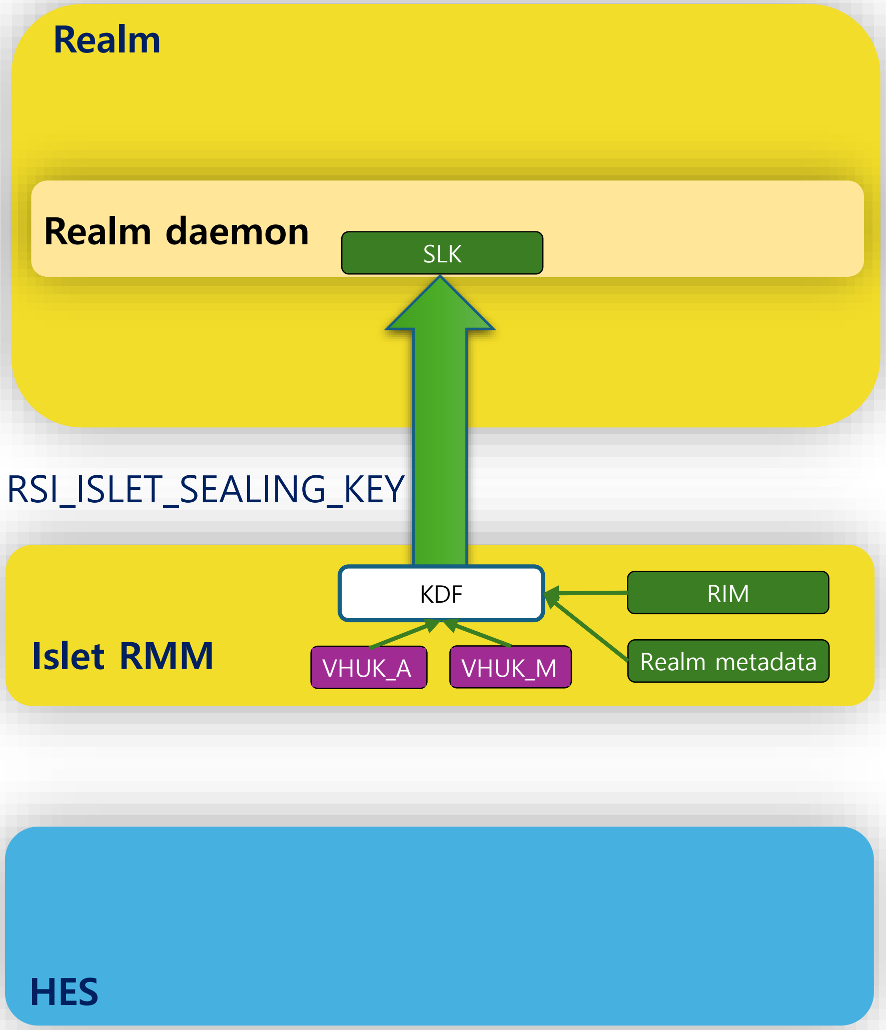
Islet RMM

During the initialization process of Realm, Islet RMM measures the Realm content.

Optionally, the KVM can provision the Realm metadata block that provides the identity information of the Realm.

Islet RMM exposes additional RMI for provisioning Realm metadata block: RMI_ISLET_REALM_SET_METADATA

Derivation of Realm Sealing Keys (SLK)



Islet RMM

Islet RMM implements additional RSI_ISLET_SEALING_KEY based on HKDF that takes additional options such as:

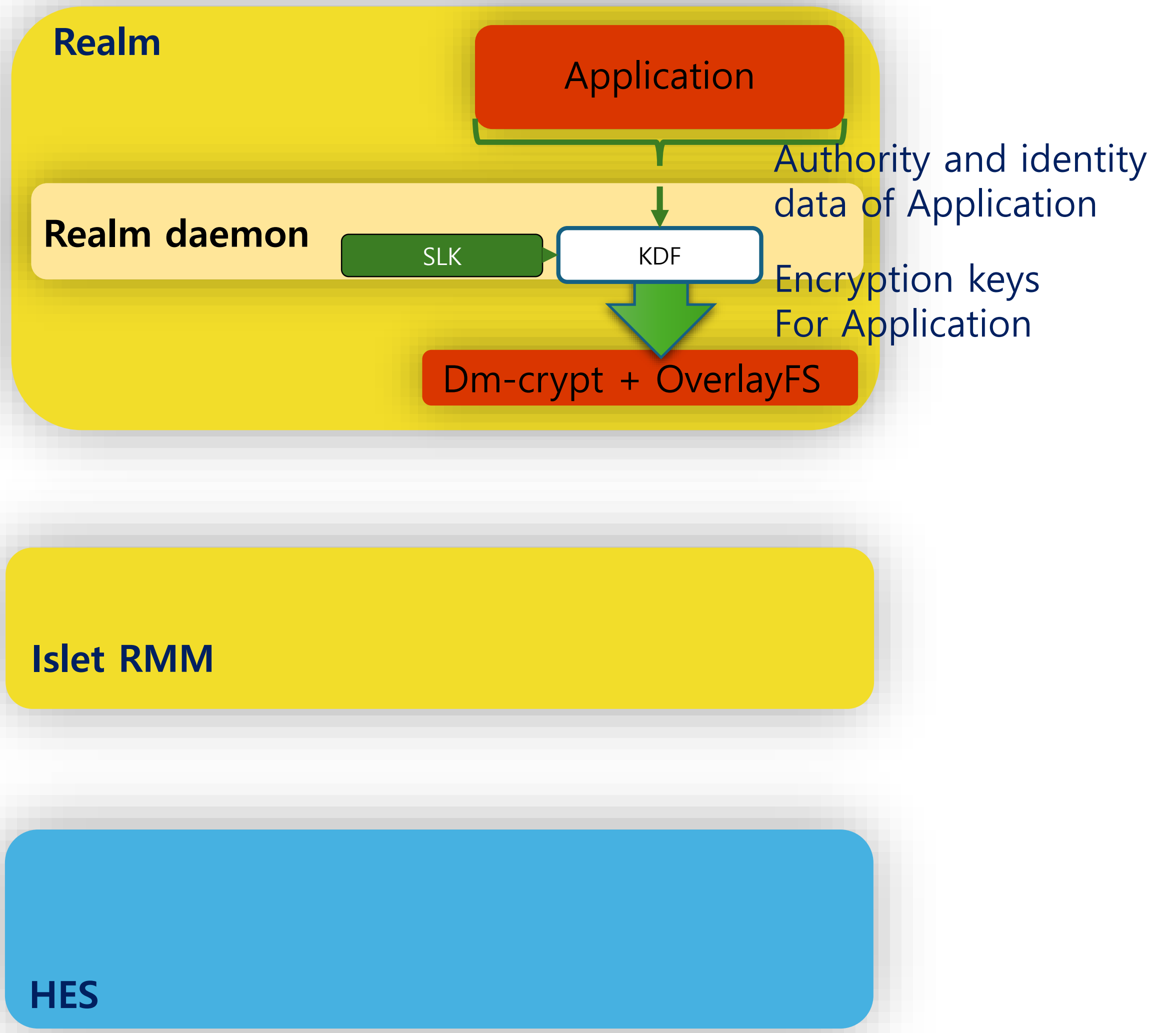
- Input Key Material selection (VHUK_A or VHUK_M)
- Use RIM measurements as a key material
- Use Realm identifier as a key material
- Use Security Version Number

By default VHUK_A is used as an Input Key Material (IKM). Realm Public Key (from metadata) and Realm Personalization Value (RPV) are always used as inputs.

If SVN is provided in options, it is included as a key material only when the SVN of the running realm is greater or equal of the SVN provided in the option.

If RMM is not provisioned with the metadata, the function derives the sealing key from the VHUK, RIM and RPV.

Derivation of secure storage encryption keys



Realm daemon

Realm daemon doesn't use the Sealing Key (SLK) retrieved from the Islet RMM directly for encryption. SLK is used as an input key material for derivation of other symmetric keys for the purpose of disk encryption.

During the startup of an application, Realm daemon derives two disk encryption keys:

- The key used for encryption of RO partition containing the application image. This key is directly derived from SLK.
- The key used for encryption of RW partition containing the application data. This key is derived from SLK, authority and identity data of the application.

The keys are derived using HKDF.

Future work

- Rollback prevention (security hardening)
  - This would require a H/W support in a form of an authenticated data storage that keeps the version information for applications and realms (e.g. eMMC that implements Reply Protected Memory Block)
  - The current design enables implementation of rollback prevention, i.e. it provides the identity of realms and applications, version numbers (e.g. SVN)
- Migration of provisioned applications to another device
  - Both involved realms should perform mutual remote attestation. This would require an external migration agent that acts as Relying Party in RATS architecture and mediates in the attestation and migration process
  - E2EE could be utilized to securely transfer the application's data directly between two authenticated and attested devices

Thank You

Samsung Research