Runtime VM Protection By Intel® Multi-Key Total Memory Encryption (MKTME)

Kai Huang @ Intel Corporation

LINUXCON + CONTAINERCON + CLOUDOPEN
Beijing, China, 2018

intel Software

Intel OpenSource TECHNOLOGY CENTER

Legal Disclaimer

Agenda

- MKTME Introduction
- MKTME Use Cases
- MKTME Enabling

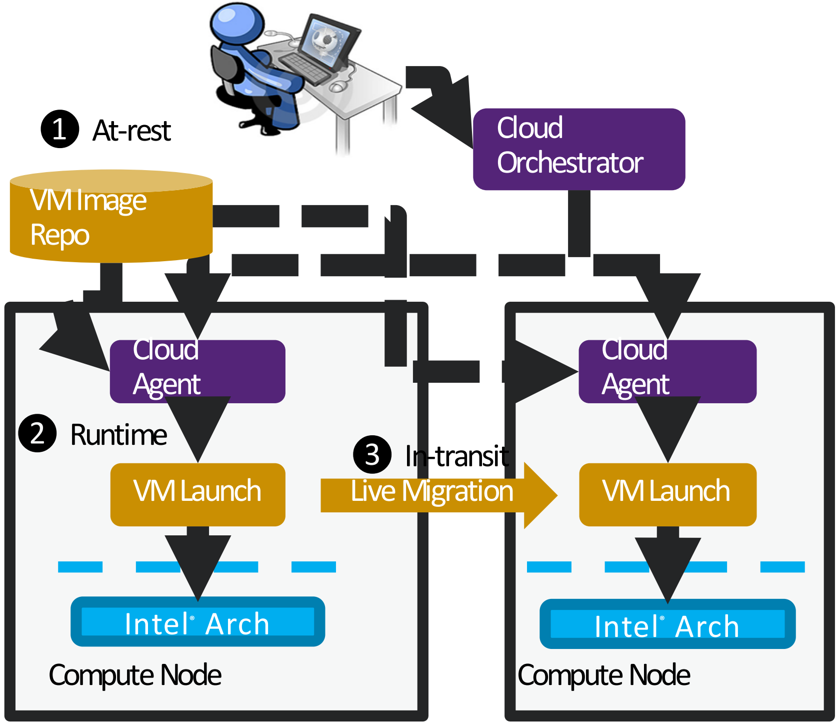
Background: Trusted VM in Cloud

VM protection by using encryption

- VM encrypted 'at-rest', 'in-transit' and 'runtime'.
- There has been existing technologies for 'at-rest' and 'in-transit' encryption
  - Qemu TLS support for live migration
  - Qemu encrypted image support
- VM runtime encryption requires **hardware memory encryption** support
  - AMD® SME/SEV
  - Intel® MKTME

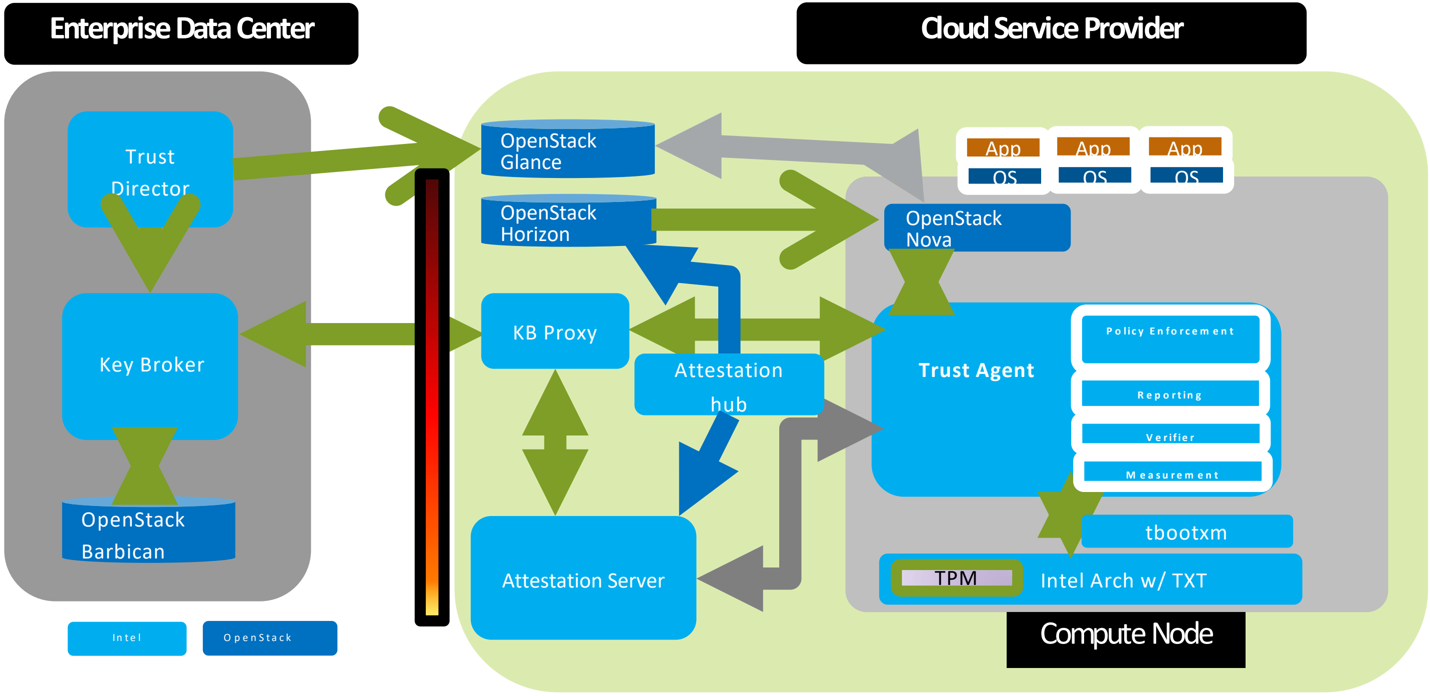
Launch VM on 'Trustiness Verified' Host

- Trusted hardware/location, etc.
- Trusted Cloud SW stack.



Typical VM Lifecycle in Cloud

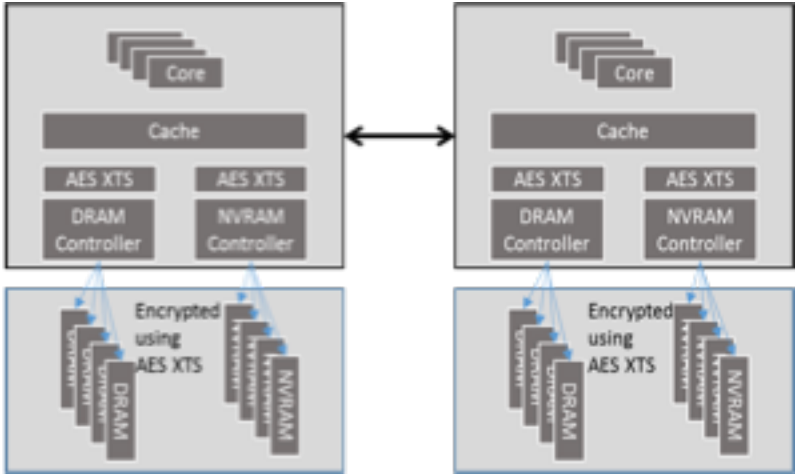
Trusted VM w/ OpenCIT -- OpenStack as Example



Intel® Open CIT helps on **Host trustiness verification**

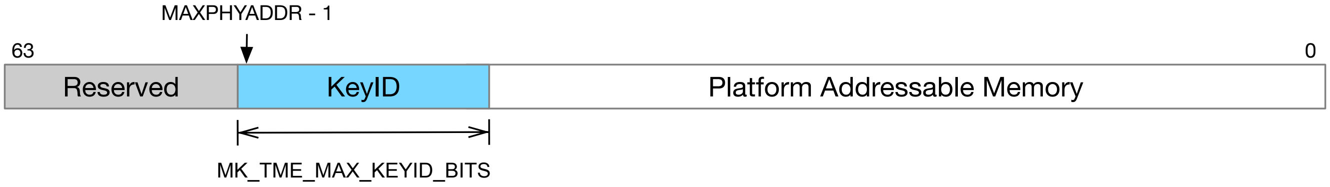
TME & MKTME Introduction

- New AES-XTS engine in data path to external memory bus.
  - Data encrypted/decrypted on-the-fly when entering/leaving memory.
  - AES-XTS uses physical address as “tweak”
    - Same plaintext, different physical address -> different ciphertext.
- TME (Total Memory Encryption)
  - Full memory encryption by TME key (CPU generated).
  - Enabled/Disabled by BIOS.
  - Transparent to OS & user apps.
- MKTME (Multi-key Total Memory Encryption)
  - Memory encryption by using multiple keys.
  - Use upper bits of physical address as keyID (see next)



MKTME KeyIDs

- Repurpose upper bits of physical address as KeyID as shown below.
  - Reduces useable physical address bits.
  - Creates “aliases” of physical memory locations: different keyIDs can refer to the same page.
  - Cache-coherency is not guaranteed for the same page that accessed by different keyIDs.
    - CPU caches are unaware of keyID (still treat keyID as part of PA)
- Architecturally upto 2^15-1 keyIDs (15 keyID bits).
  - Reported by MSR. Configured by BIOS.
  - KeyID 0 is reserved as TME’s key (not useable by MKTME).
- New PCONFIG instruction to program keyID w/ associated key (see next)



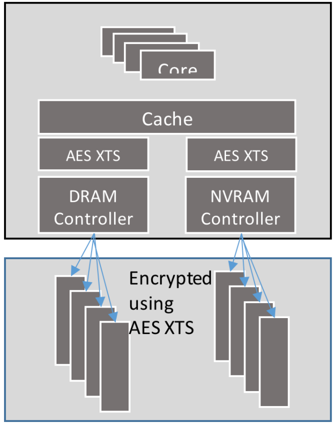
MKTME KeyID Programming Overview

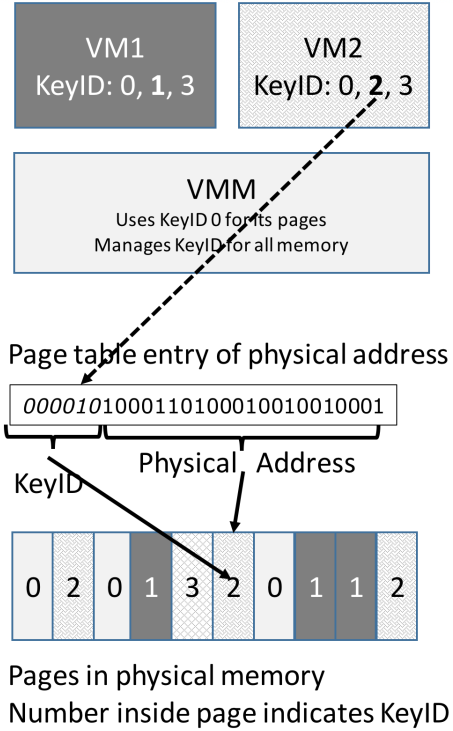
New Ring-0 instruction PCONFIG to program the KEYID and associated key

- Package scoped
- Supports programming keyID to 4 modes:
  - Using CPU generated random ephemeral key (invisible to SW)
  - Using SW provided key (tenant's key)
  - No encryption – plaintext domain
  - Clearing a key (using TME's key effectively)
- Allows SW to specify crypto algorithms
  - Only AES-XTS-128 for initial server intercept

VM Protection & Isolation With MKTME

- Protection
  - Use keyID to encrypt VM memory at runtime
- Isolation
  - Use different keyIDs for different VMs
- Software Enabling
  - For CPU access, SW sets keyID at PTEs
    - IA page table (host)
    - EPT (KVM)
  - For Device access (DMA)
    - w/ IOMMU: Set keyID to IOMMU page table
    - Physical DMA: Apply keyID to PA directly





Highlights of MKTME

Guests continue to run "*without modifications*" in MKTME domains:

- Encrypted with 1) CPU-generated ephemeral key, or 2) the one provided by API ("tenant-controlled keys")
- Virtio, including optimization (direct access to guest memory by kernel) continues to work
- Direct I/O (including accelerators, FPGA) assignment (including SR-IOV VFs) is available
- Live migration can be supported (among platforms that support MKTME)
- vNVDIMM can be supported w/ limitation (because of physical address "tweak")
  - Host DIMM configuration cannot be changed cross reboots.
  - Qemu DIMM & vNVDIMM configuration cannot be changed cross VM reboots.

Agenda

- MKTME Introduction
- MKTME Use Cases
- MKTME Enabling

MKTME Enabled Use Cases

1. ***Launch Tenant VMs with in-use protection (CPU generated keys)***

- Let CSP handle the keys
- VM image provided by CSP

2. ***Launch Tenant VMs with at-rest and in-use protection with full tenant-control***

- VM image encrypted @rest with tenant-specific keys
- Keys in tenant control
- VM memory isolation with tenant-specific keys
- Trustiness verified host
- Additional: integrity verification of VM image

Use-case Extension:

KeyID Sharing for all VMs launched by single tenant with the same tenant-key (or CPU generated key).

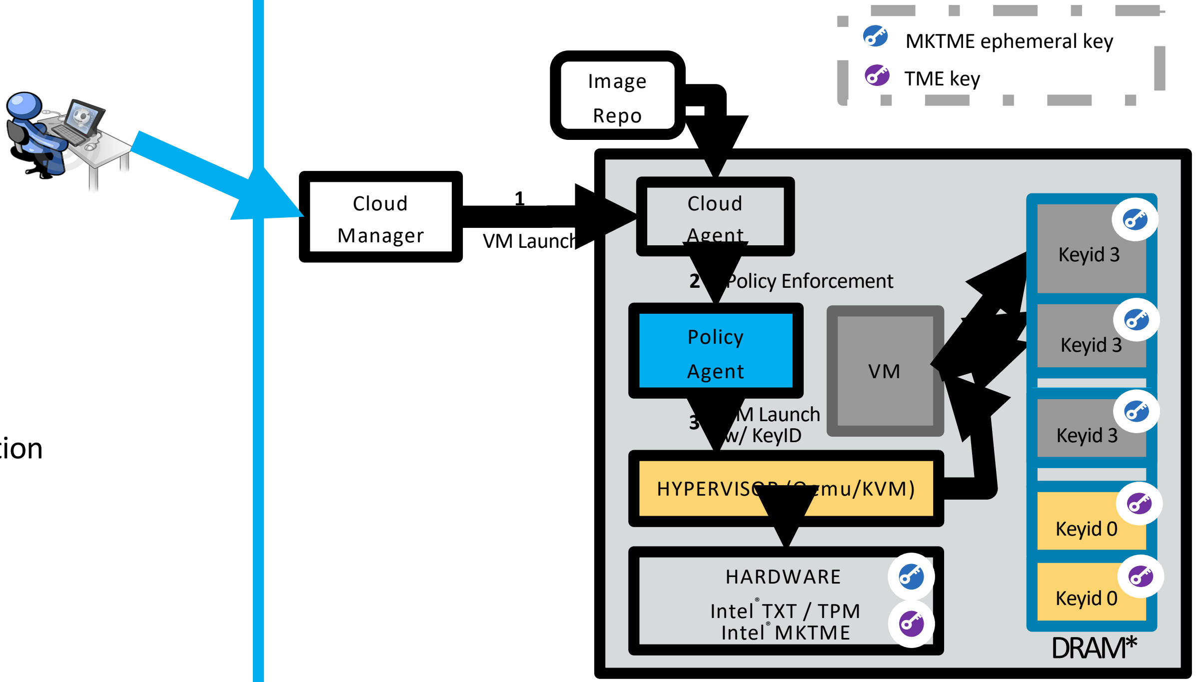
VM Launch w/ CPU Generated Keys

VM Launch w/

- CPU generated key
- CSP provided VM image

Security Properties

- w/ VM runtime protection
- w or w/o at-rest and in-transit protection
- No Host Trustiness Verification



CSP Controlled

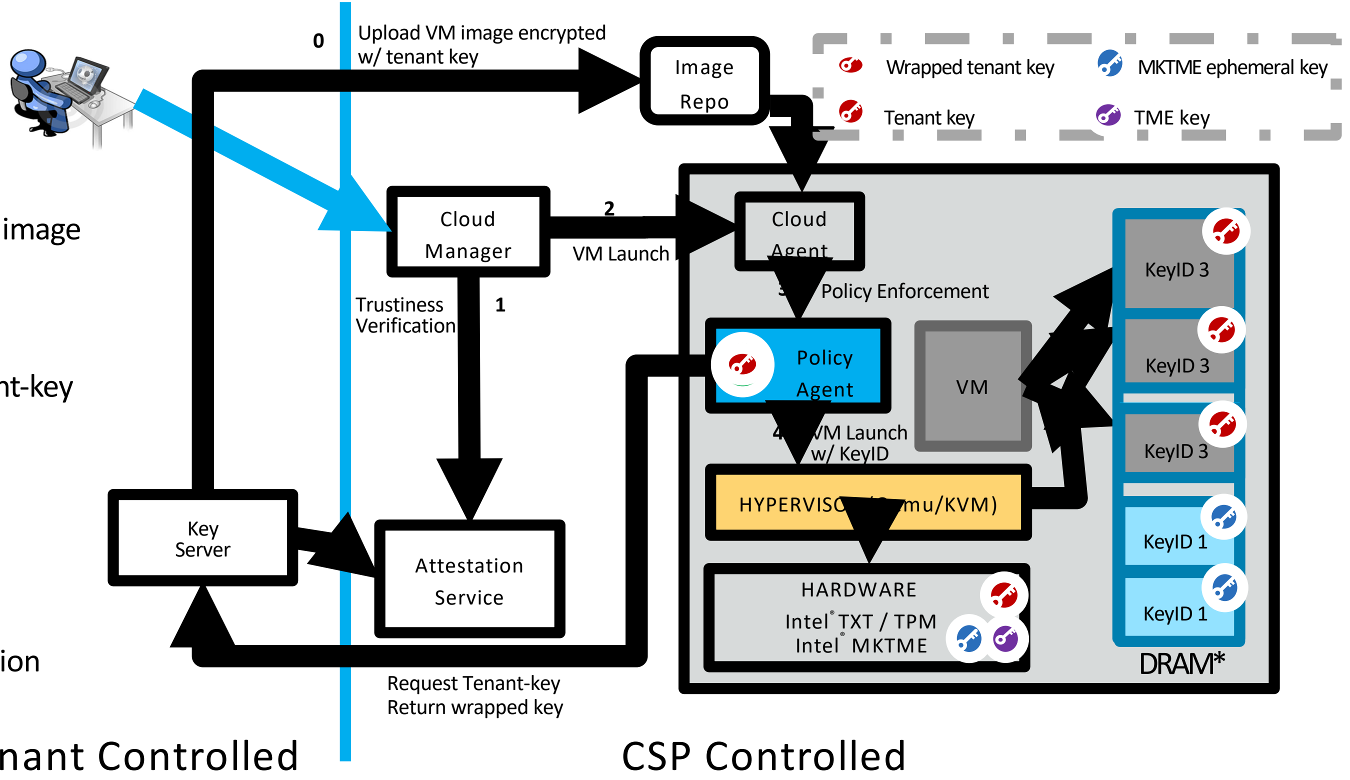
VM Launch w/ Tenant Controlled Keys

VM Launch w/

- Tenant provided key
- Tenant provided encrypted VM image
- Tenant controlled key server
- Trustiness verified host
- VM image integrity verified
- Use TPM to wrap/unwrap tenant-key

Security Properties

- w/ VM runtime protection
- w/ VM at-rest protection
- w/ or w/o in-transit protection
- w/ Host trustiness verification
- w/ VM image integrity verification



KeyID Sharing Among VMs

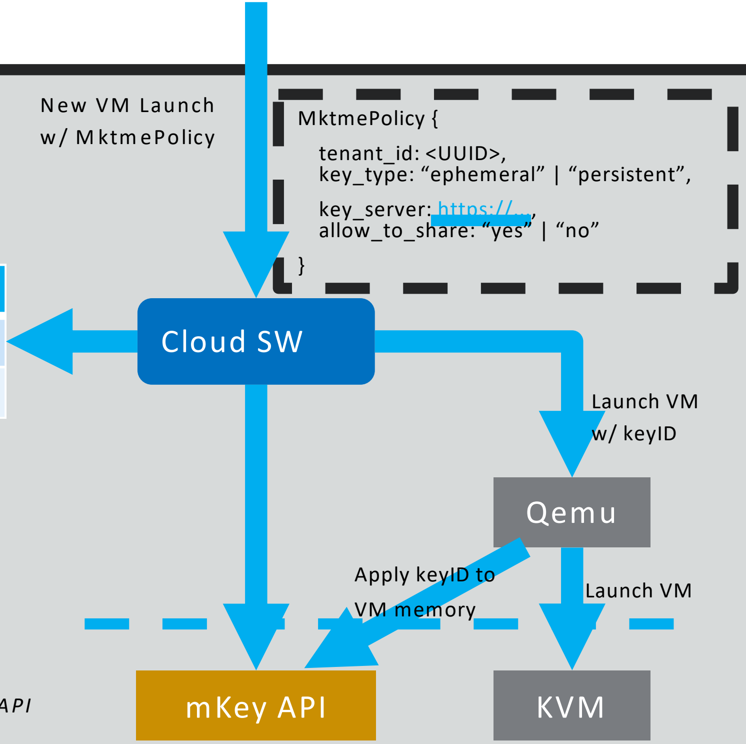
Cloud SW makes decision whether to share or not.

KeyIDPolicy	KeyID	VMs
Policy1: <tenant1, “ephemeral”>	keyID1	VM1, VM2..
Policy2: <tenant2, “persistent”, xxxxxx>	keyID2	VM3

Example: KeyID sharing is based on KeyIDPolicy: <tenant_id, key_type, tenant_key>

Cloud SW:

- *Maintains ‘KeyIDPolicy-to-KeyID’ table*
- *Makes keyID sharing decision according to the table*
- *Updates the table on VM launch and teardown*



Compute Node

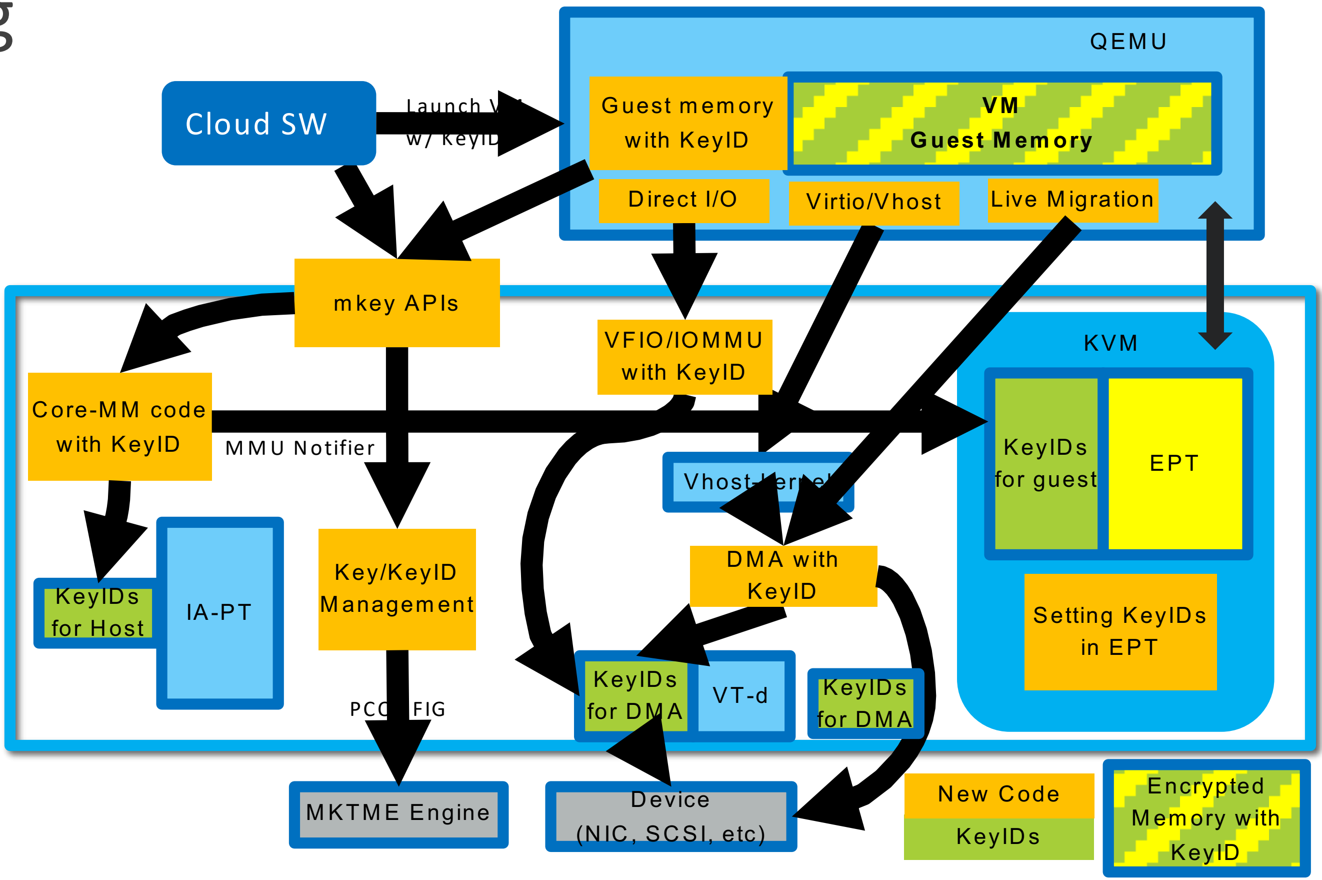
mKey API: MKTME key management API

Agenda

- MKTME Introduction
- MKTME Use Cases
- MKTME Enabling

MKTME Enabling High Level

- Host kernel
  - mkey APIs
  - Key/KeyID Management
  - Core-MM KeyID support
  - VFIO/IOMMU KeyID support
  - DMA KeyID support
- KVM
  - KeyID setup in EPT
- Qemu
  - Receive KeyID from Cloud SW
  - Apply KeyID to guest memory



MKTME Enabling Current Status

- Specification has been published [1]
- Core kernel enabling status
  - Some preliminary patches have been upstreamed
    - Feature emulation (CPUID, MSR); PCONFIG
  - Proposal of some components have been sent to community for feedback
    - Key management API: Using kernel key retention service
  - Other components WIP internally
    - Core-MM keyID support; IOMMU keyID support; DMA keyID support; ...
- KVM/Qemu enabling status
  - PoC has been done to prove MKTME actually works.
  - Depending on core kernel parts ready for formal patches.

[1] https://software.intel.com/sites/default/files/managed/a5/16/Multi-Key-Total-Memory-Encryption-Spec.pdf

THANKS