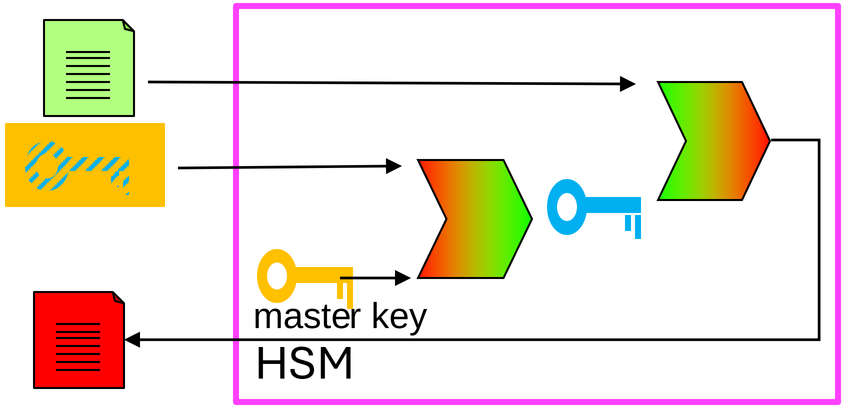
# Enabling Hardware Security Modules for Confidential Computing

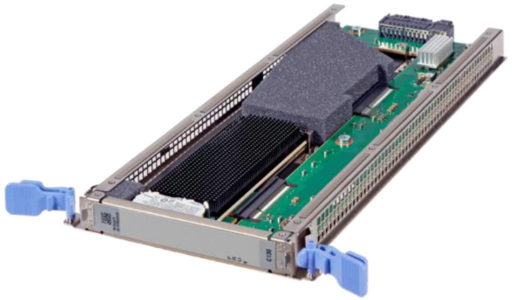
Reinhard Bündgen – buendgen@de.ibm.com

STSM, Chief Architect for Confidential Computing and Security for Linux on IBM Z and LinuxONE

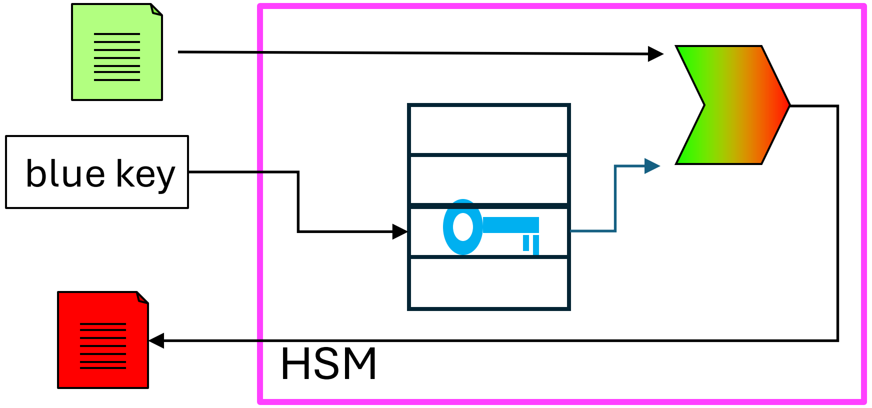
# What are Hardware Security Modules (HSMs)?

- Hardware device
- Protects keys objects:
  - HSM protected keys can only be used inside HSM
  - Plaintext values of HSM protected keys are not observable outside of HSM
- Good HSMs are tamper-respondent
  - Protect key material against physical attacks
  - Typically certified for FIPS 140-2/3 level ≥2
- Implements crypto operations
  - To generate HSM protected keys
  - Operating on HSM protected keys
- HSM protected keys are useless w/o access to HSM
  - an HSM is something you own



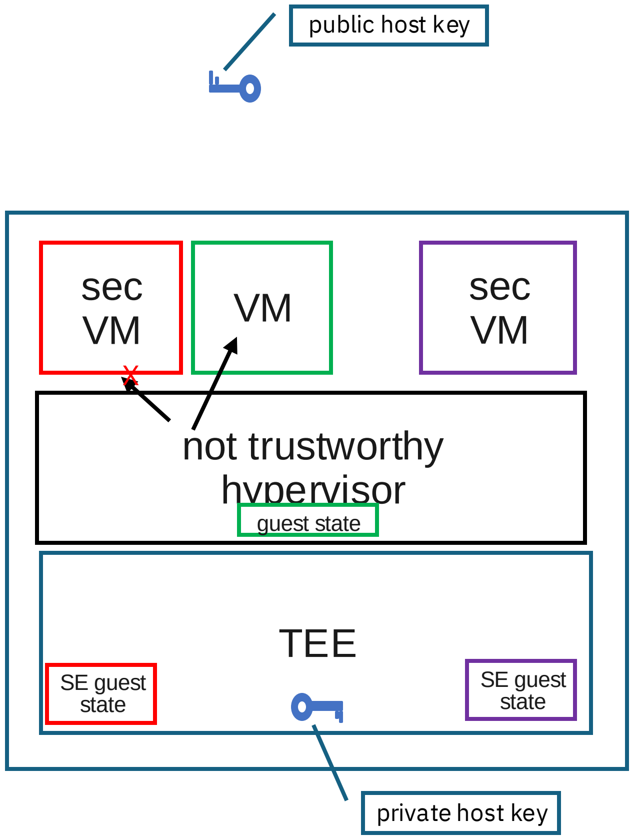







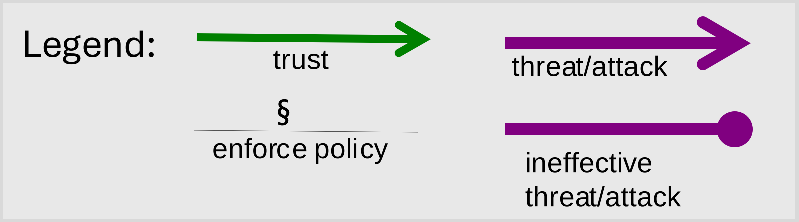
# Confidential Computing

- provides a trusted execution environment TEE for enclaves (mostly virtual machines*)
- such that the SW hosting the enclaves (e.g., hypervisor) and the administration of enclaves need not be trusted
  - hosting SW has no access to memory or registers of enclaves
  - TEE maintains sensitive state of enclave
- Examples: Intel SGX, IBM Secure Execution (SEL), AMD SEV, Intel TDX, ARM Realms




*) virtual machine (VM) aka guest

# Computation in the Cloud is a Matter of Trust

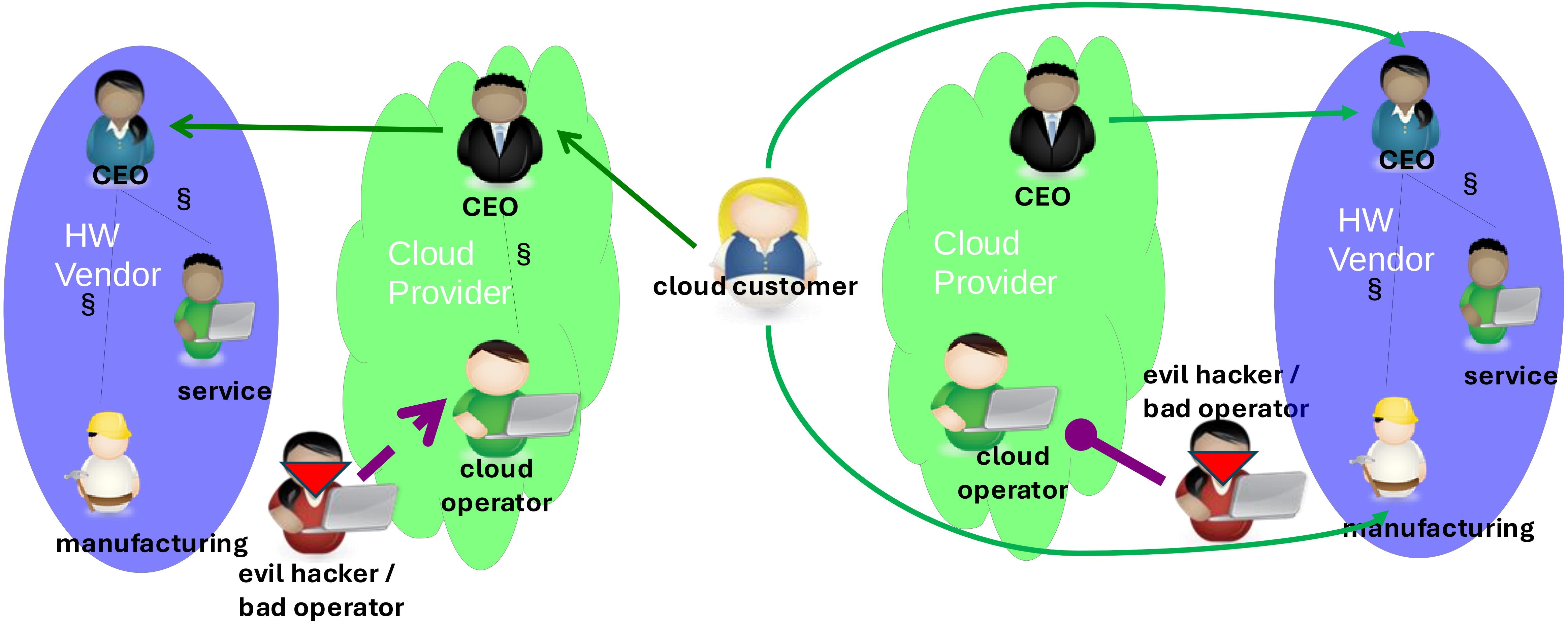



**“Traditional cloud trust model”**

- customer trusts good CEO
- good CEO enforces good policies on employees

**“Confidential Computing trust model”**

- customer trusts HW vendor



# Why do we use HSMs? What is the essence of an HSM?

(Other than checking a checkbox to satisfy a regulation)

An HSM is a means to protect cryptographic keys from being used *outside of* your system (which has access to the HSM).

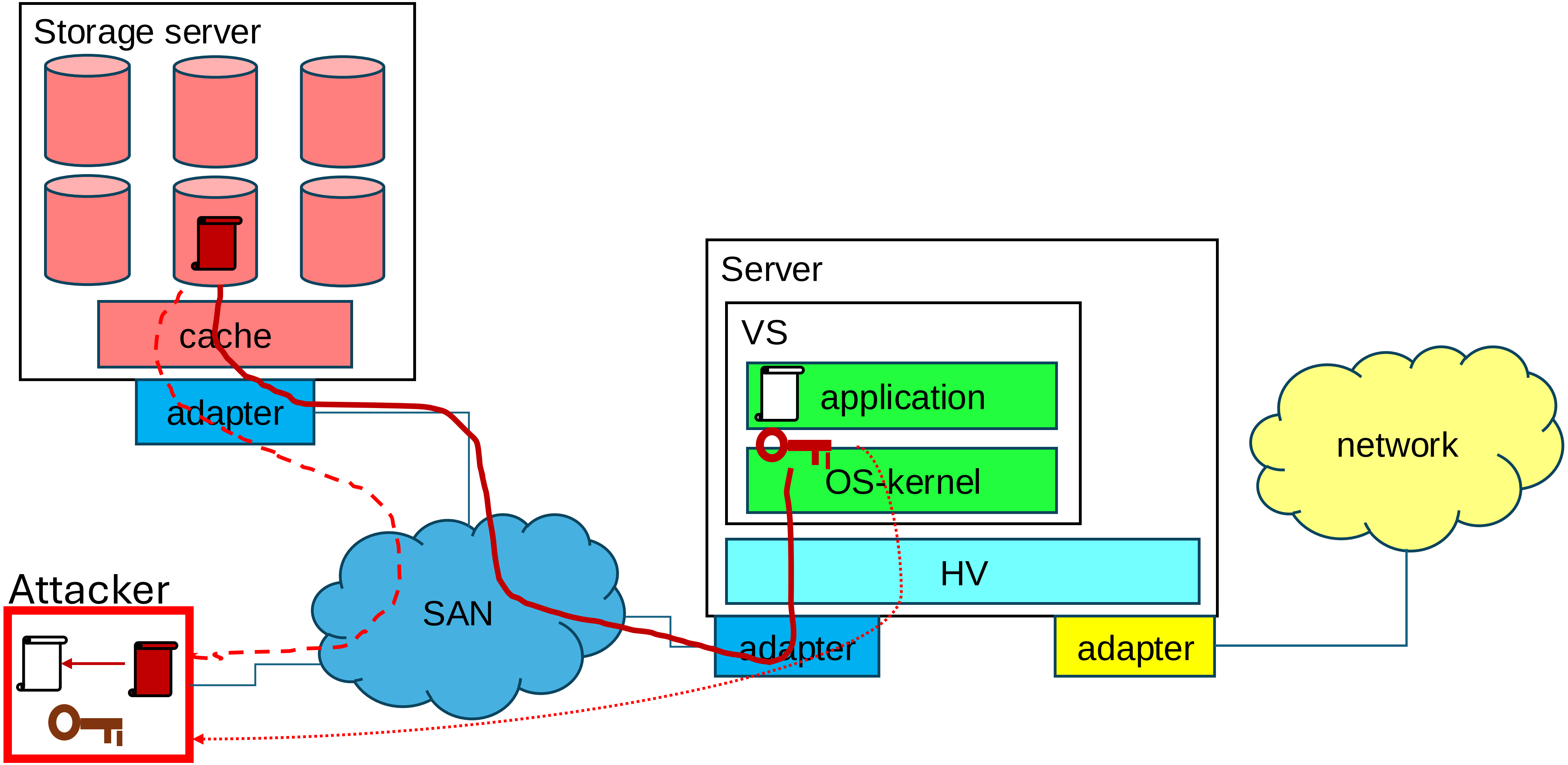
- Note, an HSM does *not* protect against an intruder in your system with root access who uses your system to observe or manipulate data.
- I.e., it protects against an **off-line attack** with your operational keys object stolen from
    - a running system (e.g., via a vulnerability like Heartbleed) or
    - a storage medium

Therefore, an HSM must be

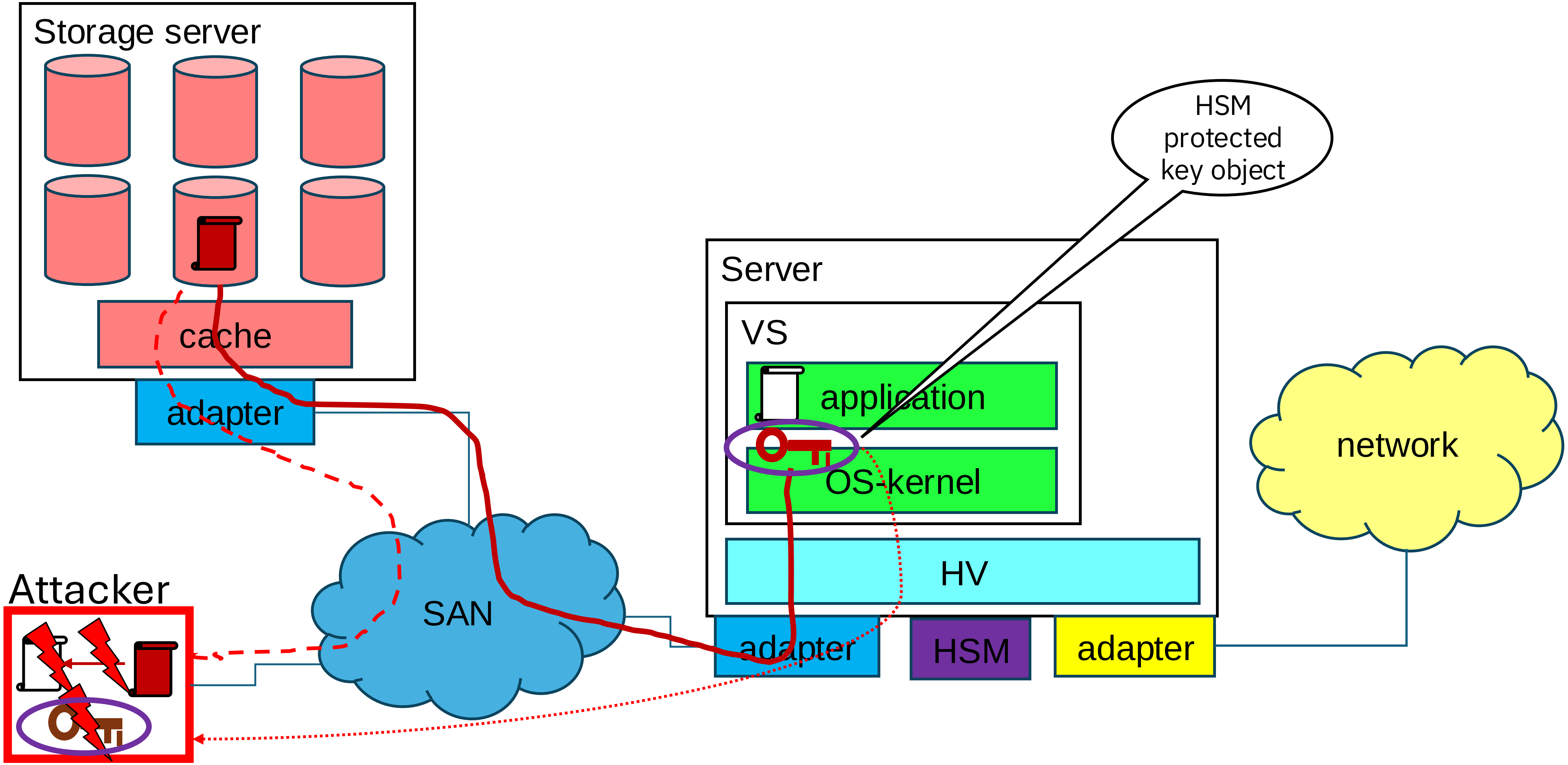
- something you own
- something that only you can use
    - remember the PIN of your credit card that protects it against lost/theft

Conclusion: **an HSM is a device that renders stolen keys useless**

# Example E2E Data Encryption: w/o HSM protection



# Example E2E Data Encryption: preventing off-line attacks with keys that cannot be stolen



# So, what is wrong with using an HSM in a non-trusted cloud?

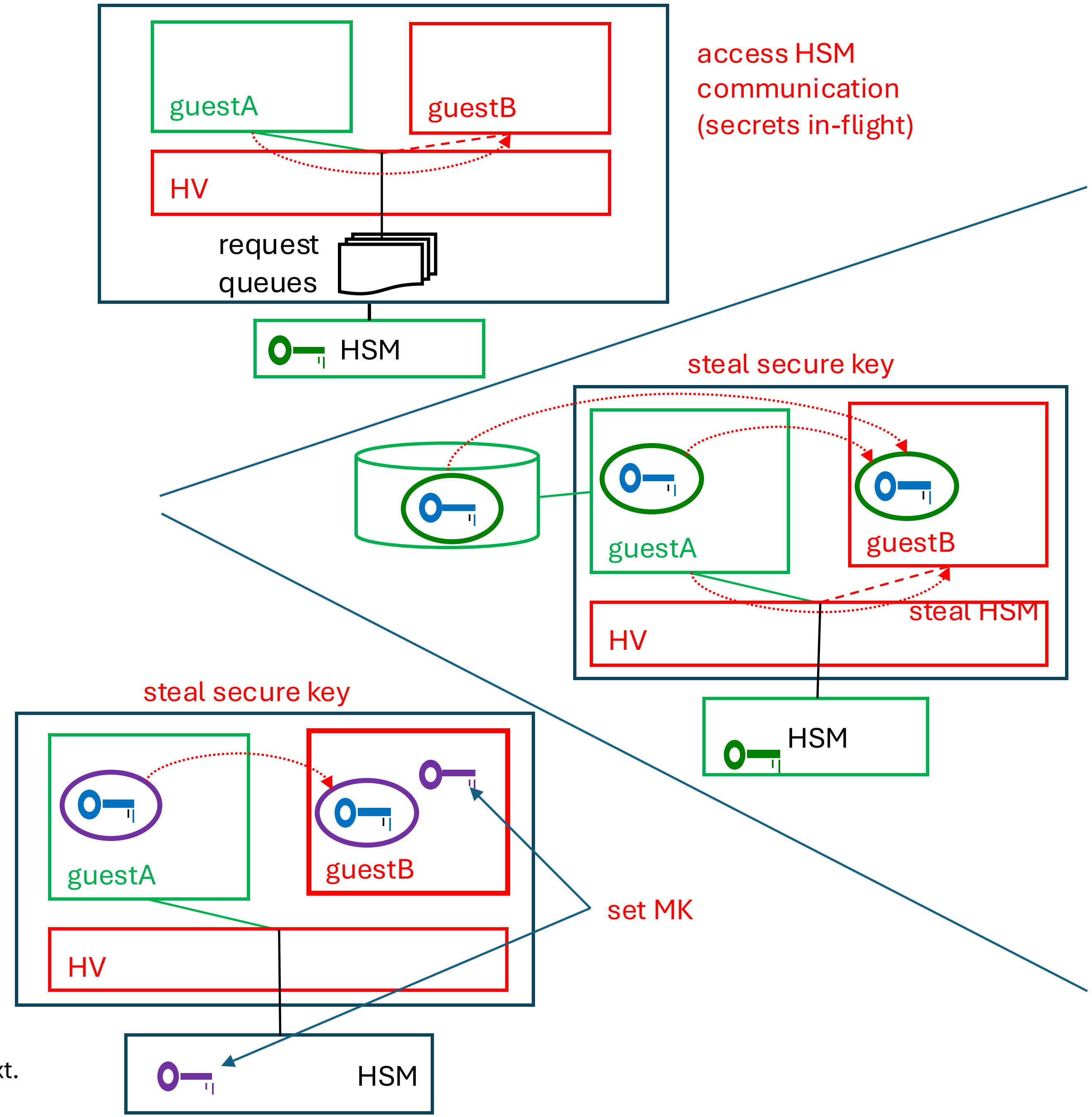
- Stealing an HSM is easy for the cloud admin: just a matter of device assignment to another system.
  - Steal both an HSM protected key and the HSM (by assigning it to the thief‘s guest) then you can use the stolen HSM protected key offline.
- So why not use a PIN?
  - Interactive entry of PINs is not reasonable on a server.
  - So, you must store the PINs somewhere, but then stealing that PIN is as easy as stealing the HSM protected key ☹
- What about network HSMs?
  - You need secrets to authenticate against the network HSM.
  - On a server you must store those secrets somewhere, but then stealing that secret is as easy as stealing the HSM protected key ☹

# Attacks on HSM usage by a guest in a virtualized environment

1. The attacker accesses HSM used by guest and reads crypto request or response
   - SIE* design allows to configure AP queue of a running guest to another component (hypervisor/HV or guest) such that a new owner can read responses of requests submitted by guest

2. The attacker both steals a secure key from guest and “steals” (i.e., configures) access to HSM to component owned by attacker
   - in a virtualized environment, an HSM is no longer something you own

3. The attacker presents an HSM to guest that is configured by attacker
   - secure keys generated on HSM are no longer secret as attacker may know HSM master key




*) SE = start interpretive execution, is the s390x instruction to enter a VM context.

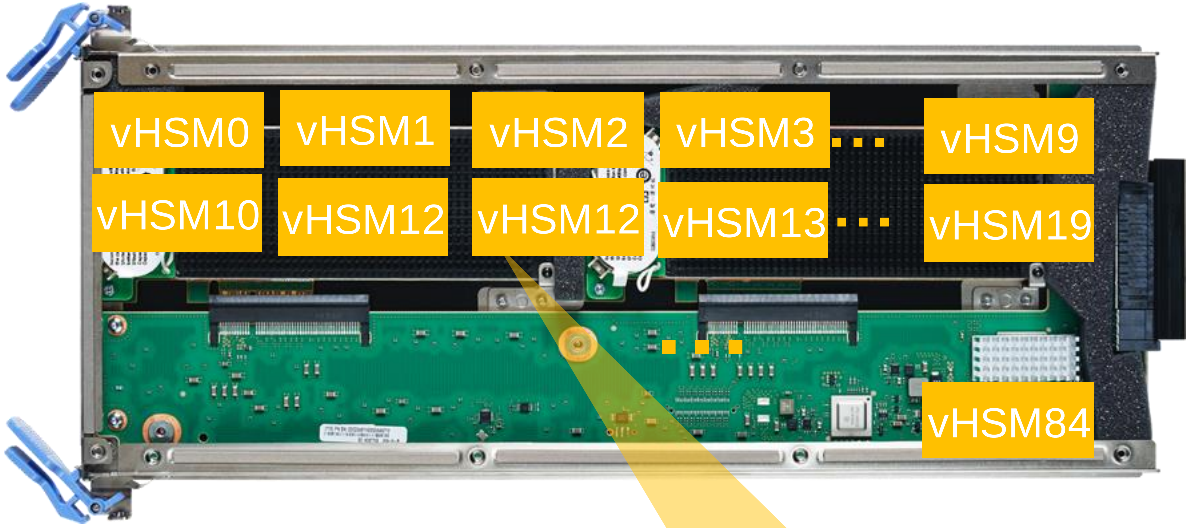
# Crypto Express Adapters

**Three different firmware loads**

– Accelerator mode

– Hardware Security Modules (HSMs):

- CCA mode
- EP11 mode




crypto adapter domain or AP queue with a master key of its own
=>
virtual HSM

**Mostly stateless HSM**

- **HSM protected keys (secure keys) are keys wrapped by a HSM master key**
- **the HSM master key cannot be extracted from the HSM**

**Adapter virtualization**

– Adapter can be partitioned into different domains of the same mode (separate master keys per domain)

– Crypto Express 8 (CEX8S) up to 85 domains

– KVM guests have passthrough access to AP queue

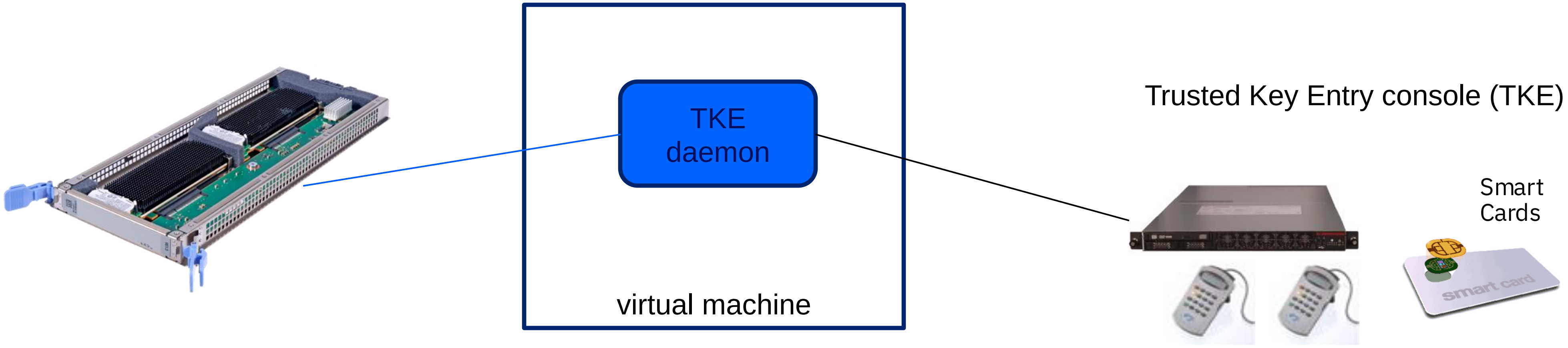
CCA mode certified as PCI HSM

FIPS 140-2 Level 4 certified

# CEX domain / master key configuration

The Trusted Key Entry console (TKE) is the management interface to configure domains in  Crypto Express adapters

– allows to take ownership of an adapter domain
  - enforces dual control
    – n out of m signatures for configuration requests
  - set master keys in Crypto Express domains
    – manages key (parts) with a build-in HSM and smart cards



# IBM Secure Execution for Linux (SEL)

**IBM z15 and IBM® LinuxONE III**

Neither HW management console nor Linux/KVM host can access

- CPU state of SEL guest*
- memory of SEL guest

A special trusted firmware component called ultravisor (UV)

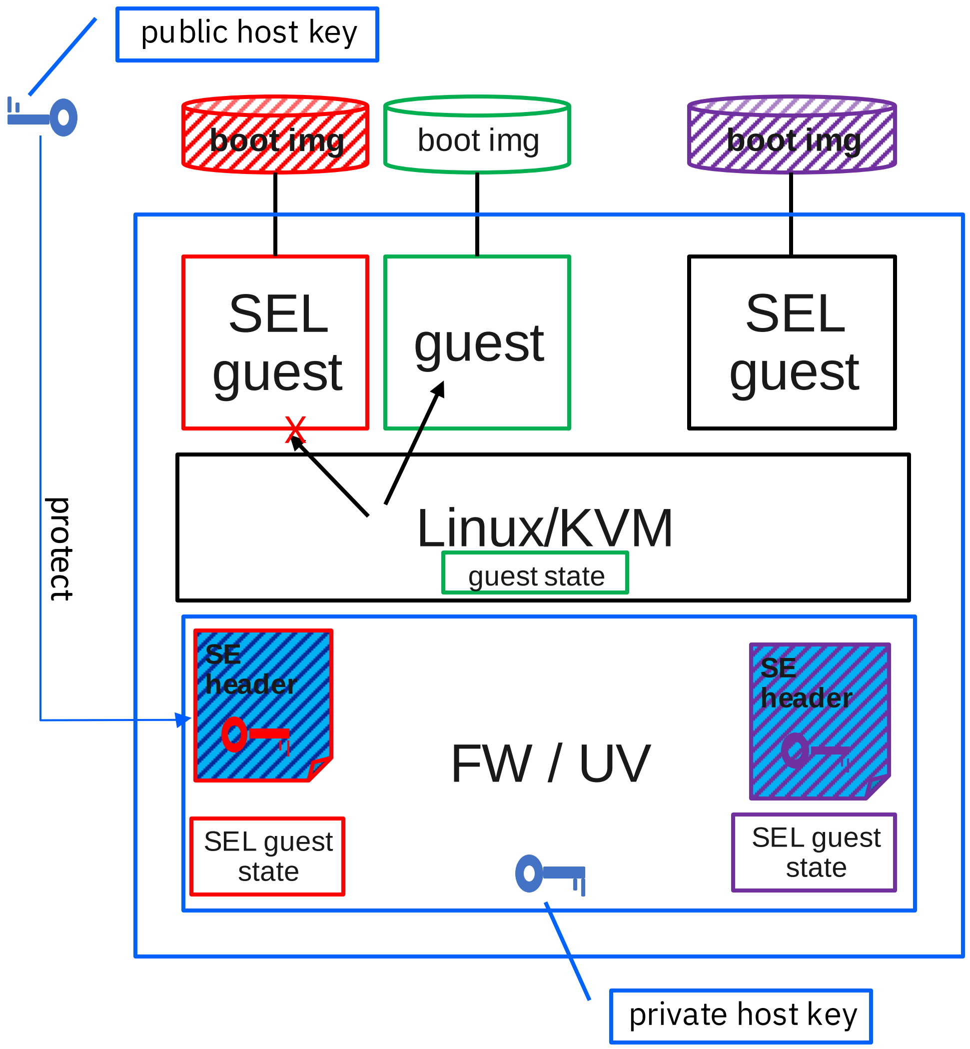
- maintains & protects the SEL guest state
- has access to the root of trust: the private host key

Image of SEL guest is integrity protected & typically encrypted

- image encryption key is passed to UV in SE-header protected using host key

**New with IBM z16 and IBM® LinuxONE 4**:

- IBM no longer keeps a copy of the private host keys.
- Support for remote attestation
  - to get trustworthy confirmation that a guest is an SE guest
  - to get a unique id of an SEL guest instance
- 04/2024: FW update to support Crypto Express accelerators and EP11 HSMs




*) guest = virtual machine

# Why is there no Crypto Express support for Secure Execution today?

The Secure Execution promise:

- the owner of a secure guest need not trust the HW admin nor the hypervisor nor the hypervisor admin

So far, a secure guest could **not** control

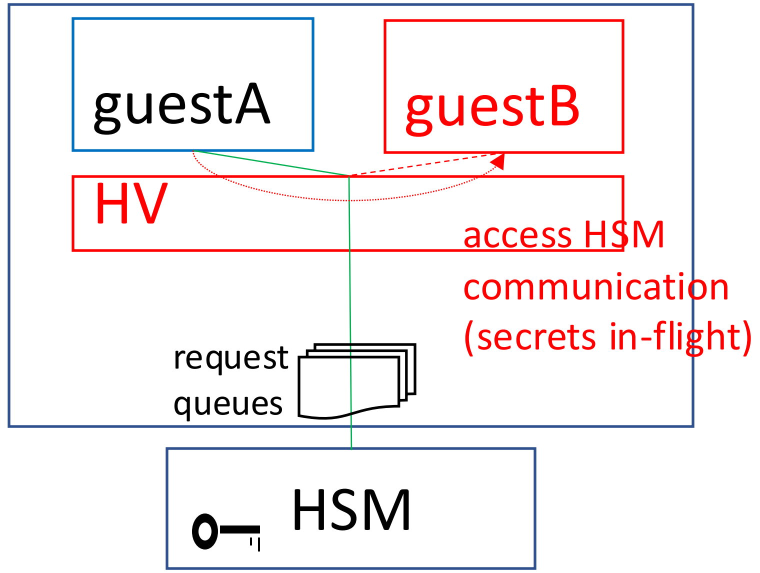
- that data exchanged with the Crypto Express domain cannot be observed by a non-trusted component
- which Crypto Express domain it is connected to
- that its Crypto Express domain is never assigned to a different guest or LPAR
- secure keys generated by a secure guest cannot be used by another guest or hypervisor

Therefore, the usage of Crypto Express adapters was disabled.

# Protection against Attack 1

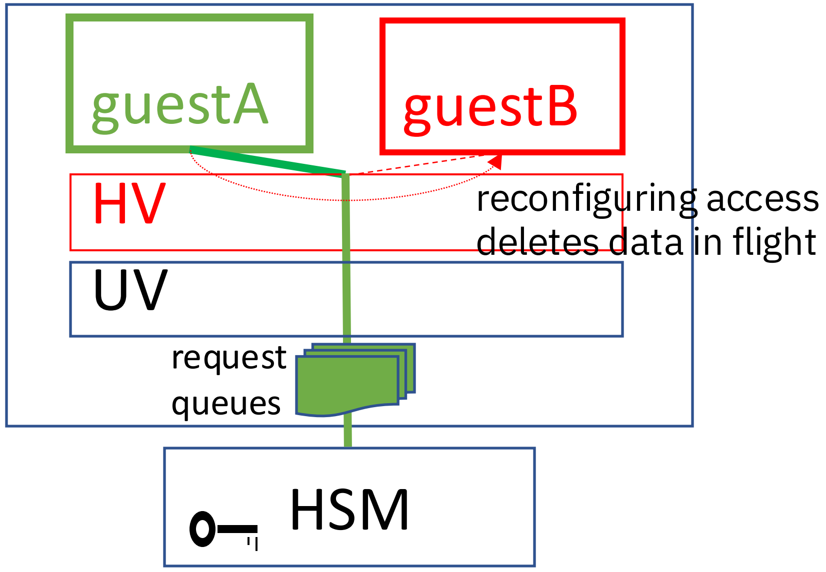
Attacker accesses HSM used by guest and reads crypto request or response

- SIE design allows to configure AP queue of a running guest to another component (HV or guest) such that new owner can read responses of requests submitted by guest




Protection: AP queue can be bound to a *running* SEL guest:

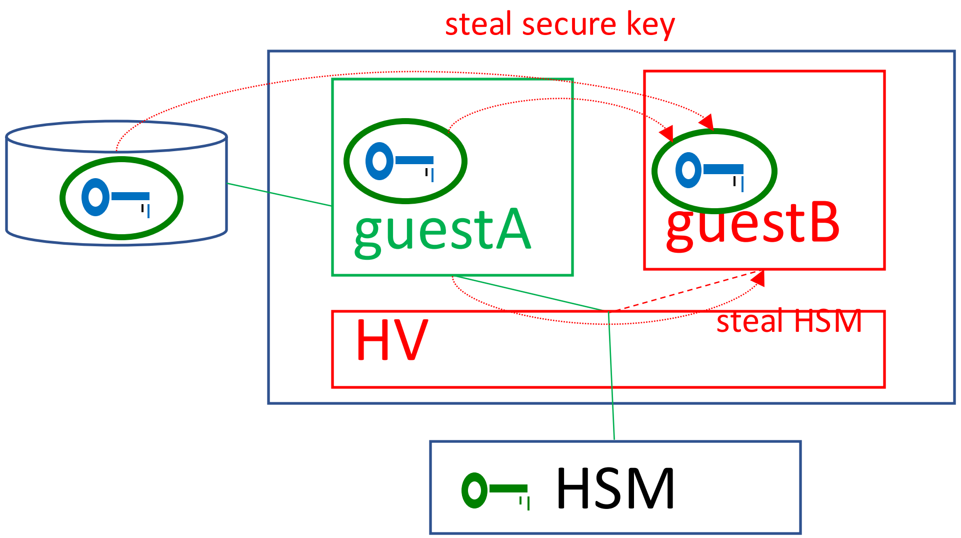
- AP queue can only be accessed by **bound** SEL-guest
- UV disables any communication between an SEL guest and AP queues not bound to the SEL guest
- other components (HW management consoles, HV, other guests):
  - cannot access AP queues bound to an(other) SEL-guest
  - resetting an AP queues undoes binding (implicit if the AP queue is (forcefully) configured to another component)
  - HW (re)plug triggers reset & unbinding
- Note, HSM can still be configured to another component
- Sufficient protection for accelerator



# Protection against Attack 2

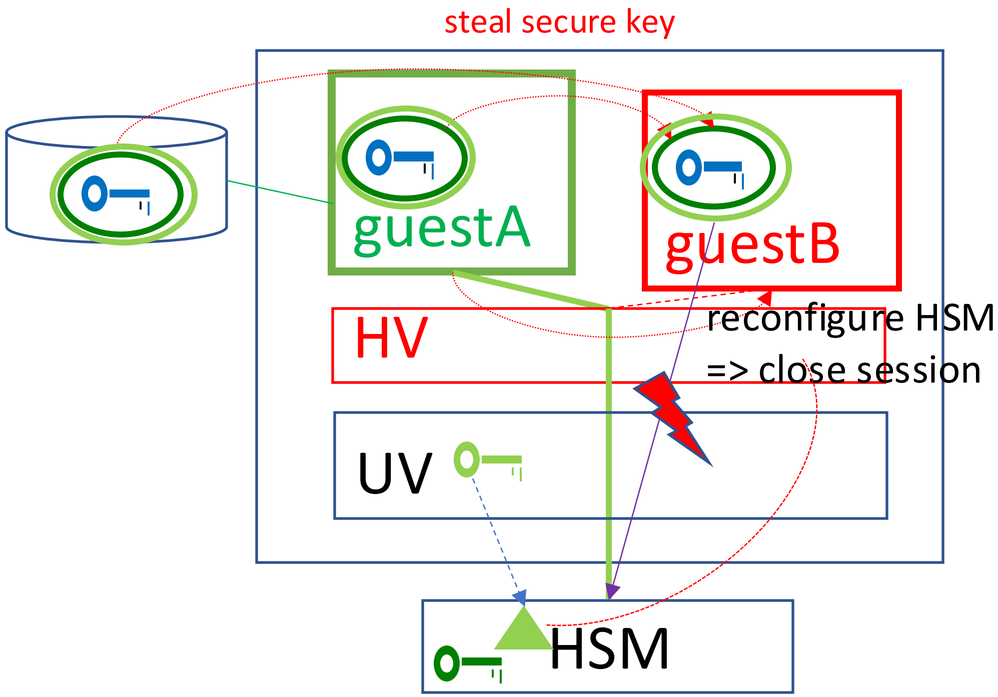
Attacker steals a secure key from guest and “steals” (i.e., configures access to)  HSM to component owned by attacker

- in a virtualized environment, an HSM is no longer something you own




Protection: UV uses association secret from SEL guest meta data and associates it with AP queue / HSM

- open a session based on association secret in HSM
- block crypto requests containing secure keys unless AP queue is associated
- all EP11 sessions opened by the SEL guest become child sessions of the session bound to the association secret
- all keys generated by an associated HSM are bound to the (child) session based on the association secret
- An AP queue will be deassociated (sessions will be closed) when the AP queue is unbound or its HSM is plugged
- Note, HSM management requests (including MKVP queries) work w/o association



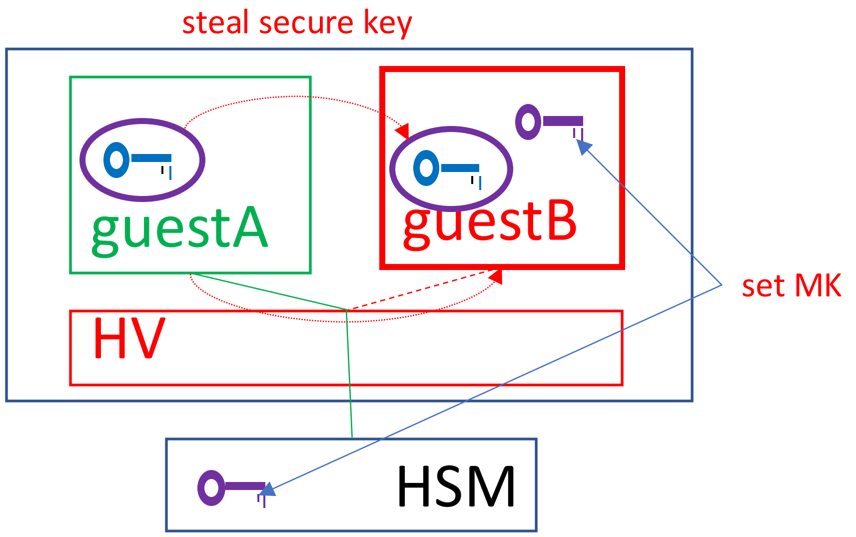
# Protection against Attack 3

## Attacker presents an HSM to guest that is configured by attacker

- secure keys generated on HSM are no longer secret as attacker may know HSM master key
- Note, if EP11 domain admins are trusted this attack is only possible while the domain is in the zeroized state

## Protection

- SEL guest can control association:
  - before associating an AP queue, the MKVP* (SN, certificate) of an HSM must be verified
    - e.g., see `sys/bus/ap/devices/<XY>.<DDDD>/mkvps`
  - association (& binding) gets lost with every HSM HW configuration event
- SEL guest program must check MKVP of each newly generated secure key
  - with openCryptoki v3.19 and later, ep11 tokens can be configured with an expected MKVP, failing all creations (generate, unwrap derive) of secure keys including an unexpected MKVP
  - zkey validate shows the MKVP of a key

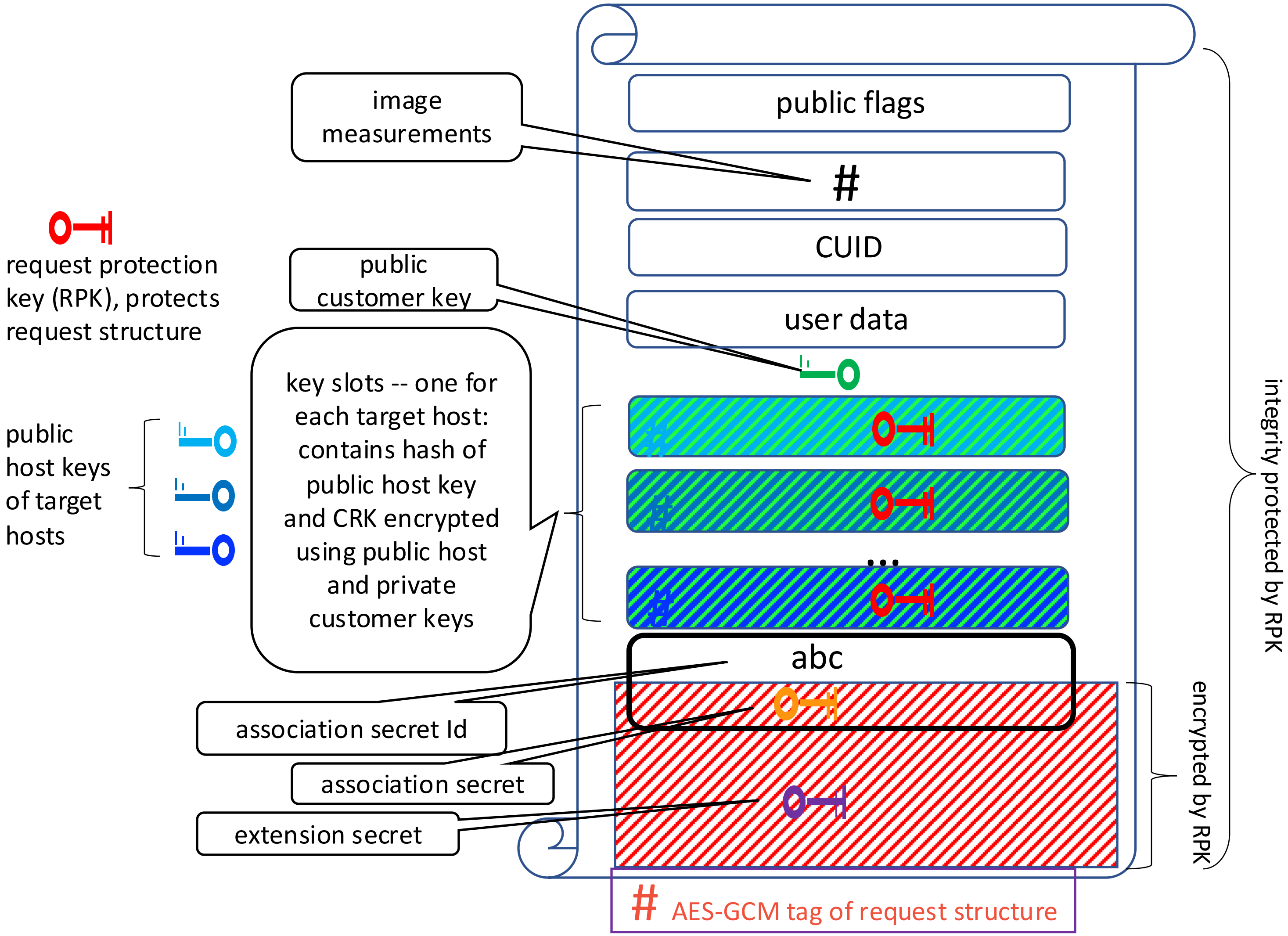



*) HSM master key verification pattern

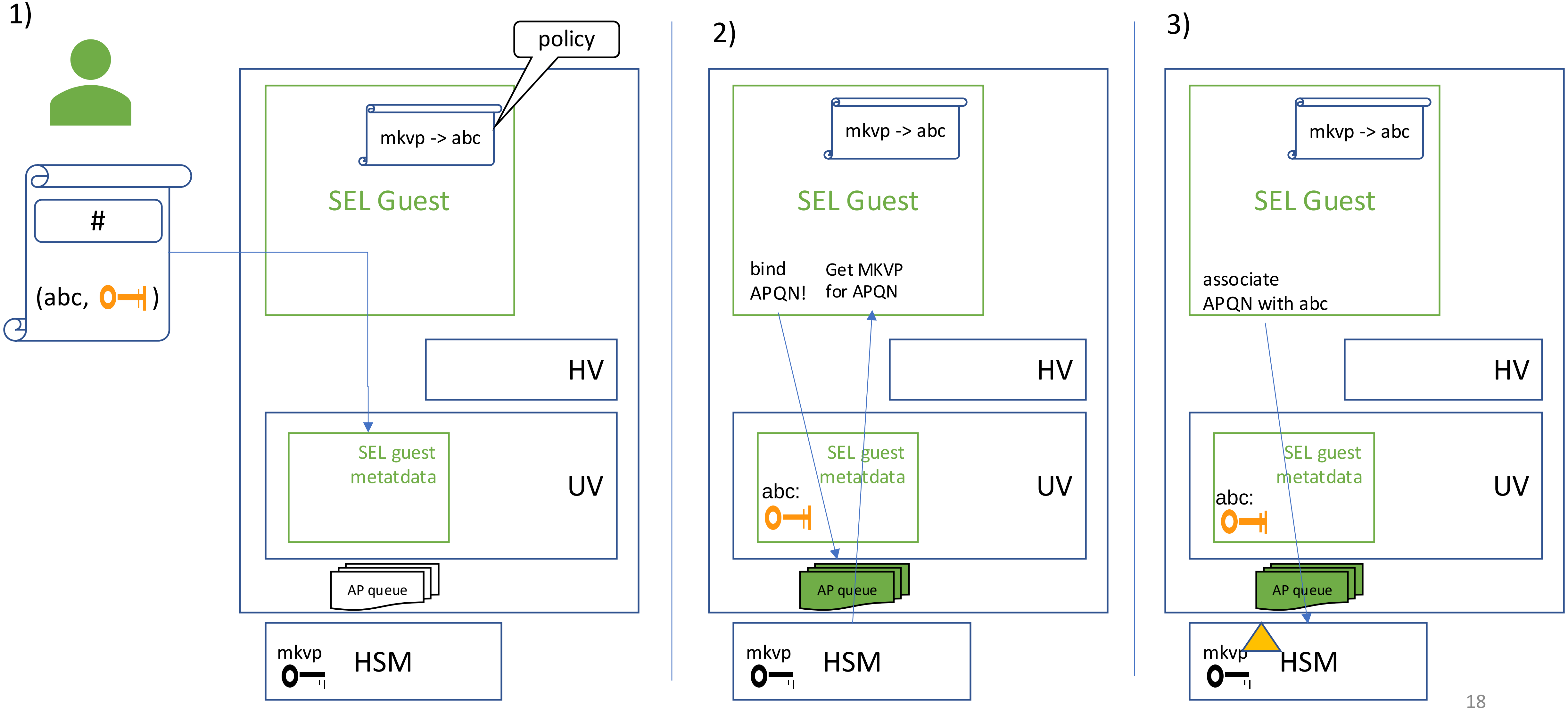
# Adding association secrets to UV protected meta data of a running guest?  -- Part 1

- A secret used to associate an AP queue is called association secret
- **It must never be in plaintext inside the memory of a secure guest**
- It can be submitted to the UV via an interface callable by a secure guest using an add-secret request structure that must be prepared by the owner of the guest
- There are safeguards to avoid misusing an add-secret request structure.
    - CUID
    - extension secret
        - to be shared by all add-secret requests of an SEL-guest
    - user data

## Add-secret request structure

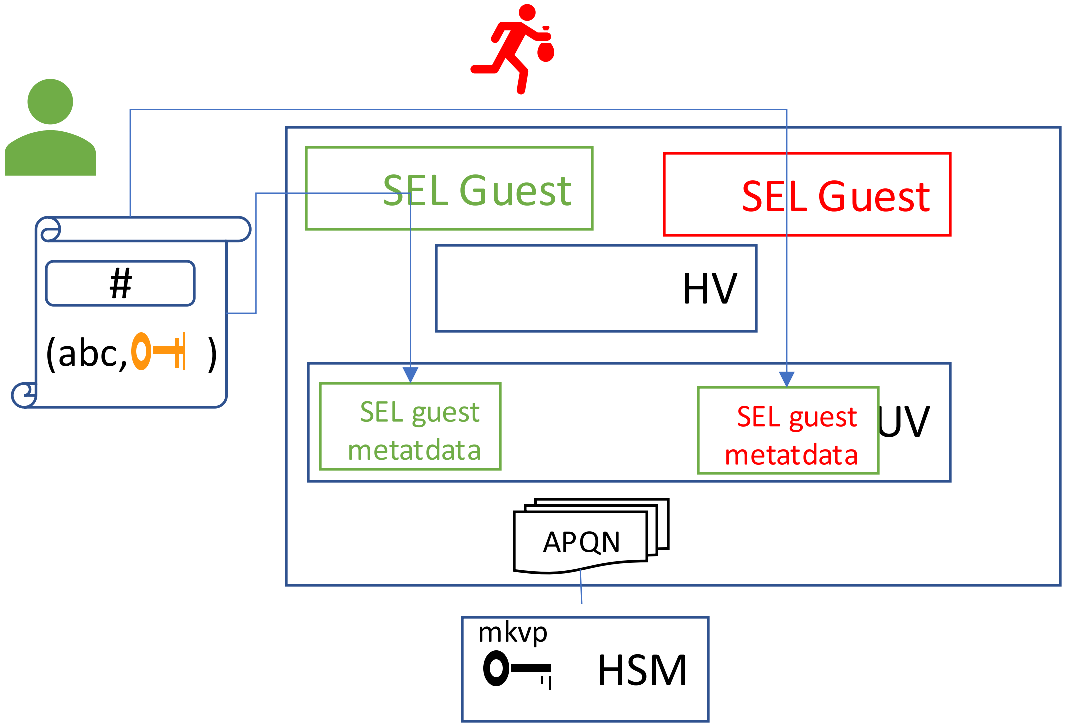


# Adding association secrets to UV-protected meta data of a running guest -- Part 2

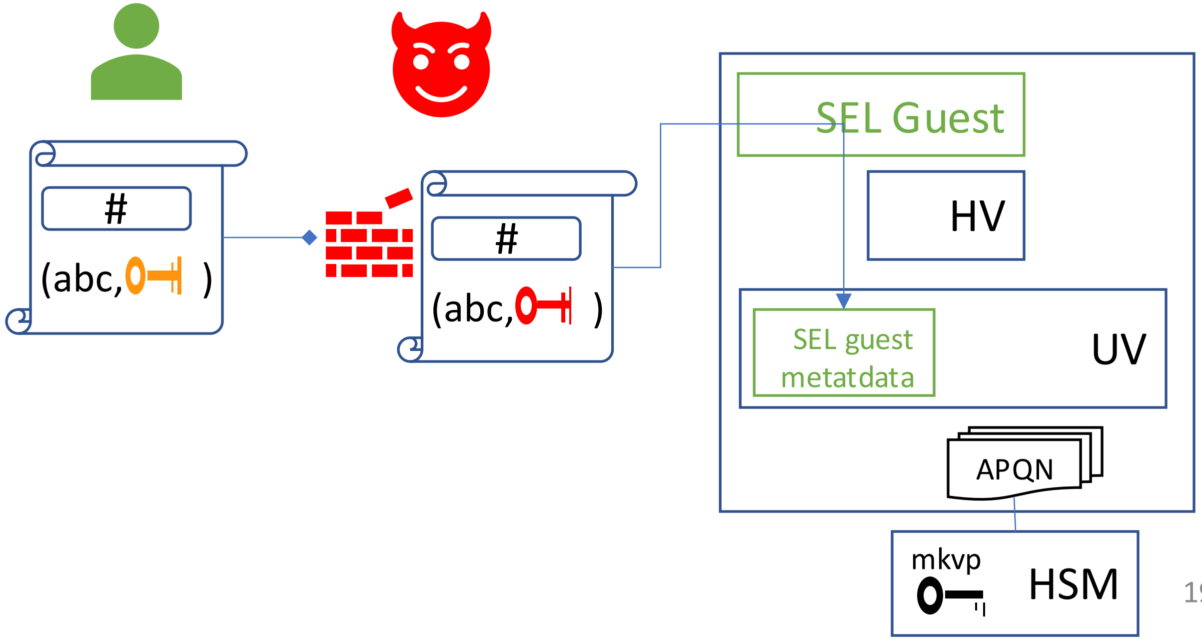


# Possible misuses of add-secret requests

- an attacker may steal an add-secret request and try to use it in an attacker's secure guest




- an attacker may trick a secure guest (owner) to use the attacker's add-secret request instead of the secure guest owner's add-secret request.



# Safeguards against misuses of add-secret requests

**Use cases**

- SEL image specific to to tenant,
  - contains tenant secrets such that only tenant has privileged access to SEL guest
  - add-secret request may or may not be generated before the SEL guest is started
- SEL image is a generic image that gets personalized after a remote attestation
  - it is OK to generate add secret requests after the guest was started and attested
- SEL image is a generic image that gets personalized with contract data including the tenant's certificate and association secrets
  - the add-secret request must be available before the SEL guest is started

Personalization of a generic confidential VM:

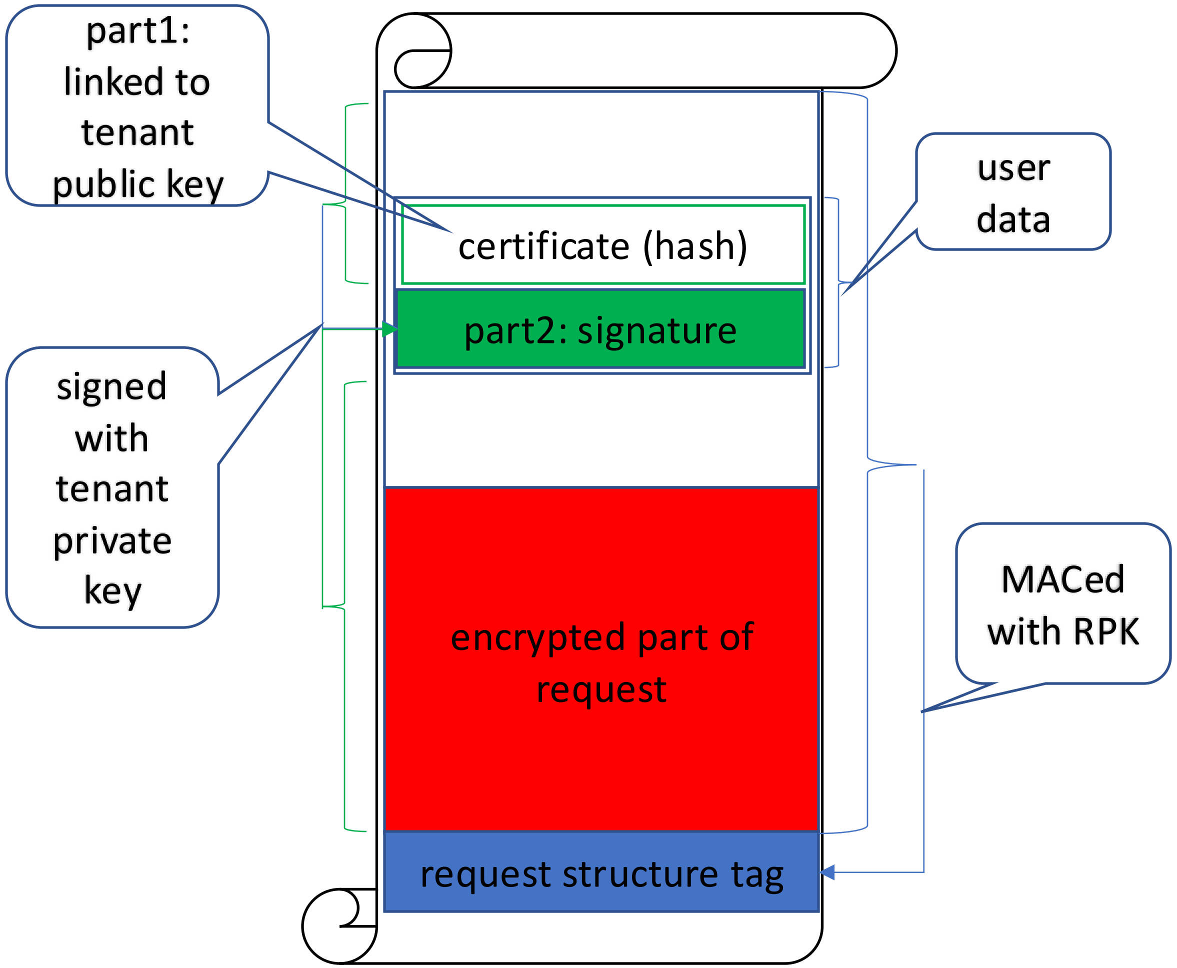
- the image of confidential VM does not contain tenant secrets
- the image may contain image vendor secrets
- the image may support remote attestation
- tenant can securely install root secrets (dm-crypt keys and/or ssh keys) to enforce exclusive (privileged) access to the confidential VM

**Safeguards**

- image and SEL header digests (ALD, PLD, TLD, SEHT):
  - Add-secret request must only be used with secure guest booted from a specific SEL image.
  - a specific behavior of the SEL guest is guaranteed during an initial phase
- extension secret
  - All add-secret requests used by a secure guest must share the extension secret.
- extension secret with CCK option
  - Owner of SEL guest image / SEL-header and add-secret request creator must share a secret.
  - genprotimg option `--enable-cck-extension-secret`
  - Not useful for generic SE guests.
- CUID
  - Add-secret request can only be used with specific guest instance.
  - the CUID can be queried with an attestation request
  - Guest must be running before the add secret request can be created.
- user data
  - Allow guest code to determine eligible add-secret requests.
  - useful for generic guest images (e.g., with personalization based on attestation or contract)
  - See next slide.
- lock secrets
  - UV call to disable guest to submit further add-secret requests to UV.
  - may be called before guest transitions from a restricted stage to a more open stage

# Binding an add-secret request to tenant certificate

- use user data in add secret request as follows:
  - part 1: optional data, e.g. digest of tenant certificate
  - part 2: signature of add-secret request with the exception of the bytes to store this signature and the MAC tag of the add-secret request using the tenant private key
- program to submit request structure to UV
  - verifies part 1 of user data to be a digest of the tenant certificate from the contract
  - verifies part 2 of user data to be a valid signature
  - only if both verifications succeed submit request to UV
- see `--user-data` and `--user-sign-key` options and `verify` sub-command options of the pvsecret tool.
- before general I/O is opened to SE guest call lock secret store UVcall



# Linux tools for HSM support for SEL

https://github.com/ibm-s390-linux/s390-tools

- kernel
  - uv DD
    - misc device node extended to submit add-secret requests to UV
  - ap/zcrypt DD
    - sys fs extensions for binding and association
    - unchanged: device node to submit crypto requests
- zcrypt tools
  - s390-tools/zconf/zcrypt
    - lszcrypt
      - show AP queue status
    - chzcrypt
      - bind & associate AP queues

- SEL image generation
  - s390-tools/genprotimg
    - genprotimg
- add-secret request handling
  - s390-tools/rust
    - pvsecret create|add|verify|lock
      - create: create request
      - add: submit request to UV
      - verify: verify user data
      - lock: prevent further secret submission
    - pvapconfig
      - apply a policy describing with association secret belongs to which MKVP

# pvapconfig – AP passthrough policies

pvapconfig associates association secrets installed in the UV with APQNs based on

- AP type (accelerator or EP11)
- master key verification patterns (EP11 only)
- association secret ID or name (SHA256 hash of name = ID)

```
# my AP config for my very secure SE guest

# we'd like to have one accelerator
- name: my Accelerator
  mode: Accel
  mingen: CEX8

# a pair of EP11 AP queues with same master key and same secret id
# but on different crypto cards as backup pair for my application
- name: my EP11 APQN 1
  mode: EP11
  mkvp: 0xdb3c3b3c3f097dd55ec7eb0e7fdbcb93
  serialnr: 93AADFK719460083

- name: my EP11 APQN 2
  mode: EP11
  mkvp: 0xdb3c3b3c3f097dd55ec7eb0e7fdbcb93
  serialnr: 93AADHZU42082261
  secretid: 0x546869732069732061207665727920736563726574207365637265742069642e
```

# Conclusion

- HSMs must not be used in environments provided by a provider that is not fully trusted, unless …
- The new Crypto Express support for Secure Execution will allow a Secure Execution guest to securely use a Crypto Express 8 adapter in accelerator or EP11 mode.

謝謝

DZIĘKUJĘ CI

TAPADH LEIBH

KEA LEBOHA

NGIYABONGA

GRÀCIES

БАЯРЛАЛАА

MISAOTRA ANAO

SALAMAT

TEŞEKKÜR EDERIM

WHAKAWHETAI KOE

KÖSZÖNÖM

DANKIE

TERIMA KASIH

DANKON

TANK

TAPADH LEAT

СПАСИБО

GRAZIE

MATUR NUWUN

ХВАЛА ВАМ

MULȚUMESC

РАКМЕТ СІЗГЕ

고맙습니다

GRAZIE

شكرا

HVALA

FAAFETAI

ESKERRIK ASKO

GO RAIBH MAITH AGAT

MAHADSANID

HVALA

БЛАГОДАРЯ

GRACIAS

THANK YOU

TEŞEKKÜR EDERIM

ТИ БЛАГОДАРАМ

DANK JE

ΕΥΧΑΡΙΣΤΩ

GRATIAS TIBI

ДЗЯКУЙ

OBRIGADO

TAK

DANKE

AČIŪ

SALAMAT

MAHALO IĀ ʻOE

TAKK SKAL DU HA

MERCI

RAHMAT

MERCI

GRAZZI

ÞAKKA ÞÉR

FALEMINDERIT

ありがとうございました

DI OU MÈSI

ĎAKUJEM

HATUR NUHUN

РАХМАТ САҒА

SIPAS JI WERE

TERIMA KASIH

CẢM ƠN BẠN

UA TSAUG RAU KOJ

WAZVIITA

ТИ БЛАГОДАРАМ

СИПОС