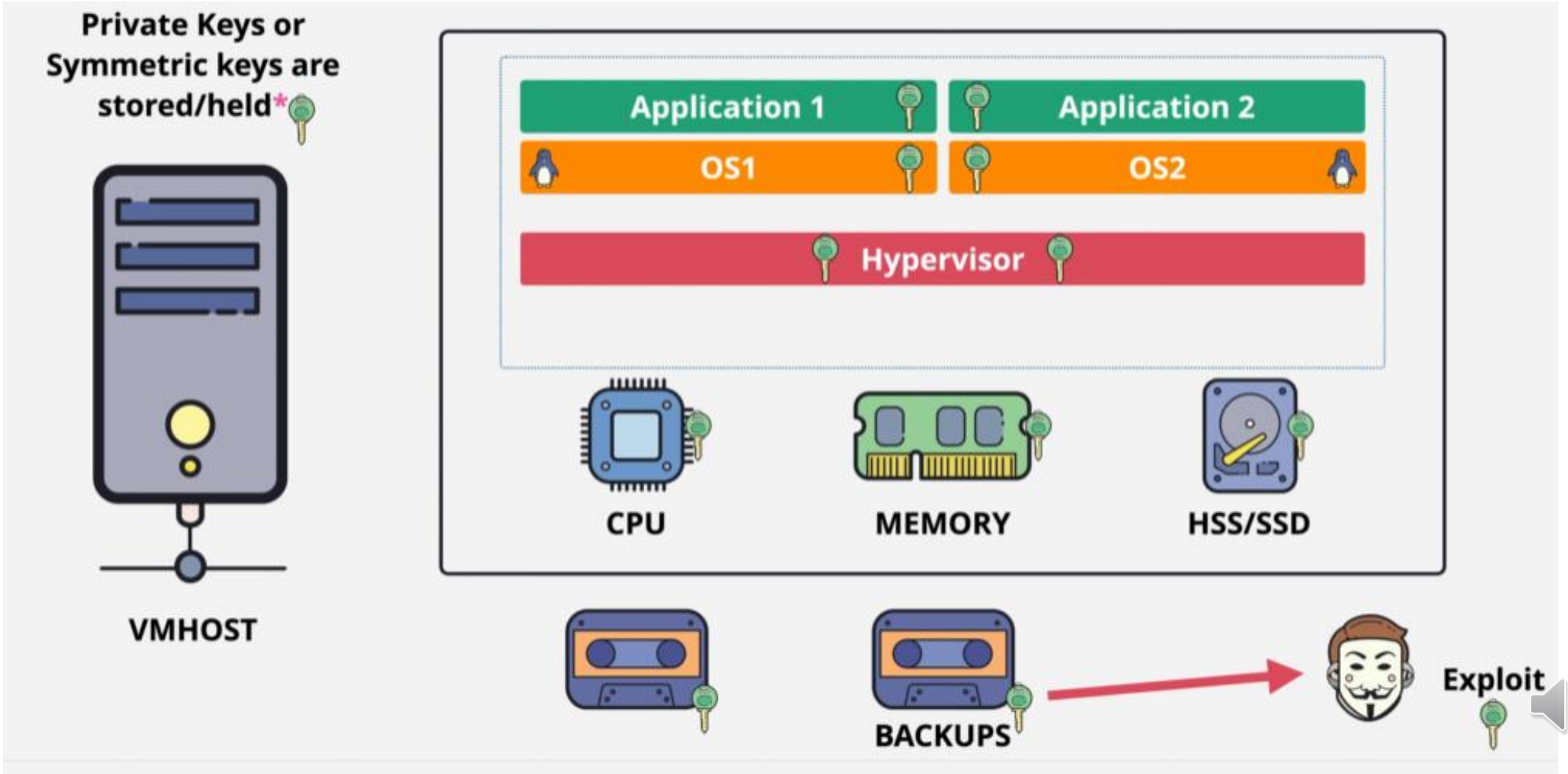
A survey of Hardware security module (HSM) and cloudHSM

Presenter: R11921A37 Yi-Ting Lee
Advisor: Professor Ming-Syan Chen
2024/03/04

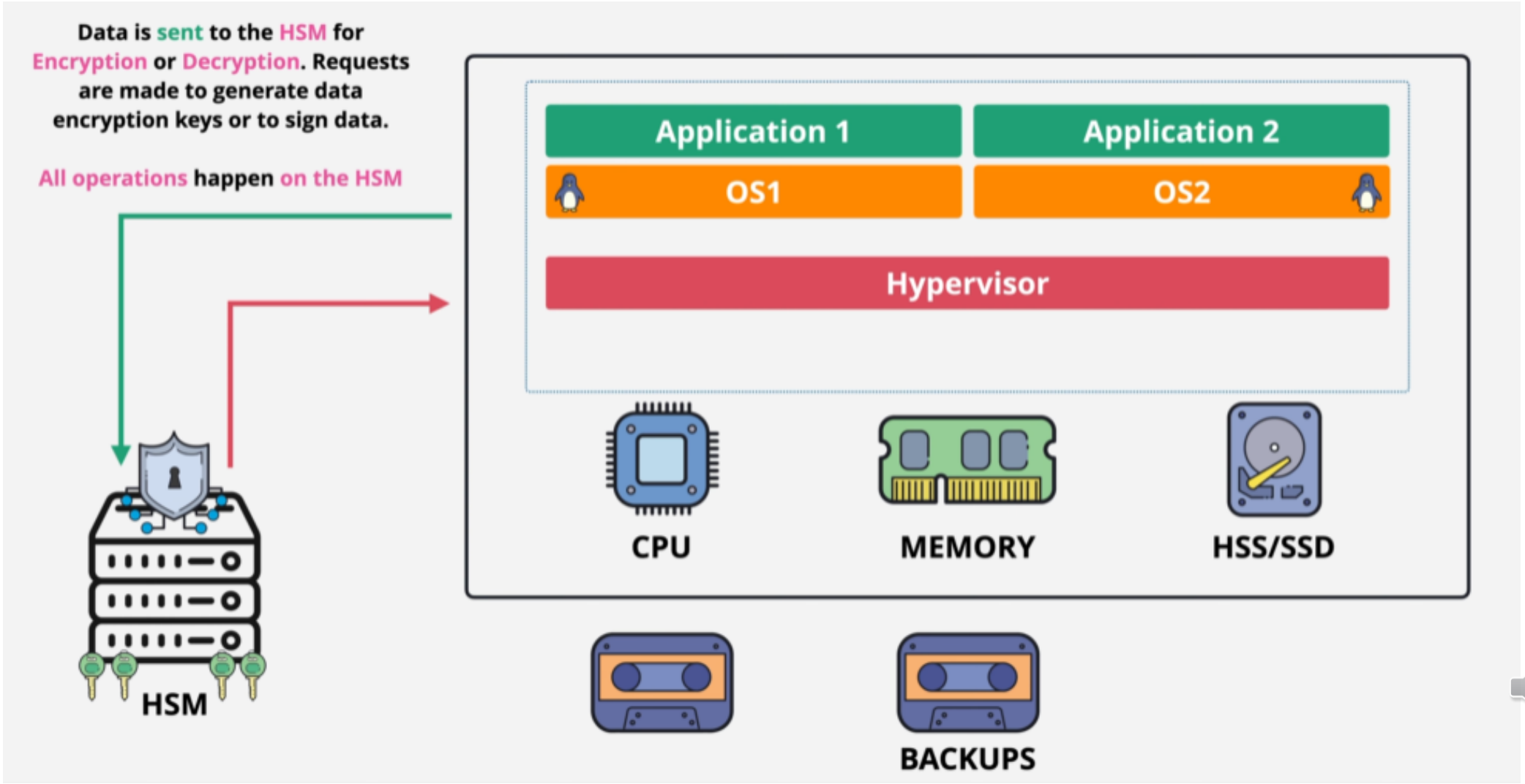
Outline

- Introduction
  - Hardware Security Module
  - Tenant Separation
  - Cloud Hardware Security Module
- Related Complementary
- Papers
  - Scalable and Secure Virtualization of HSM With ScaleTrust
    - IEEE/ACM Transactions on Networking (2022)
- Conclusions
- Insights and Future works
- Reference
- Appendix

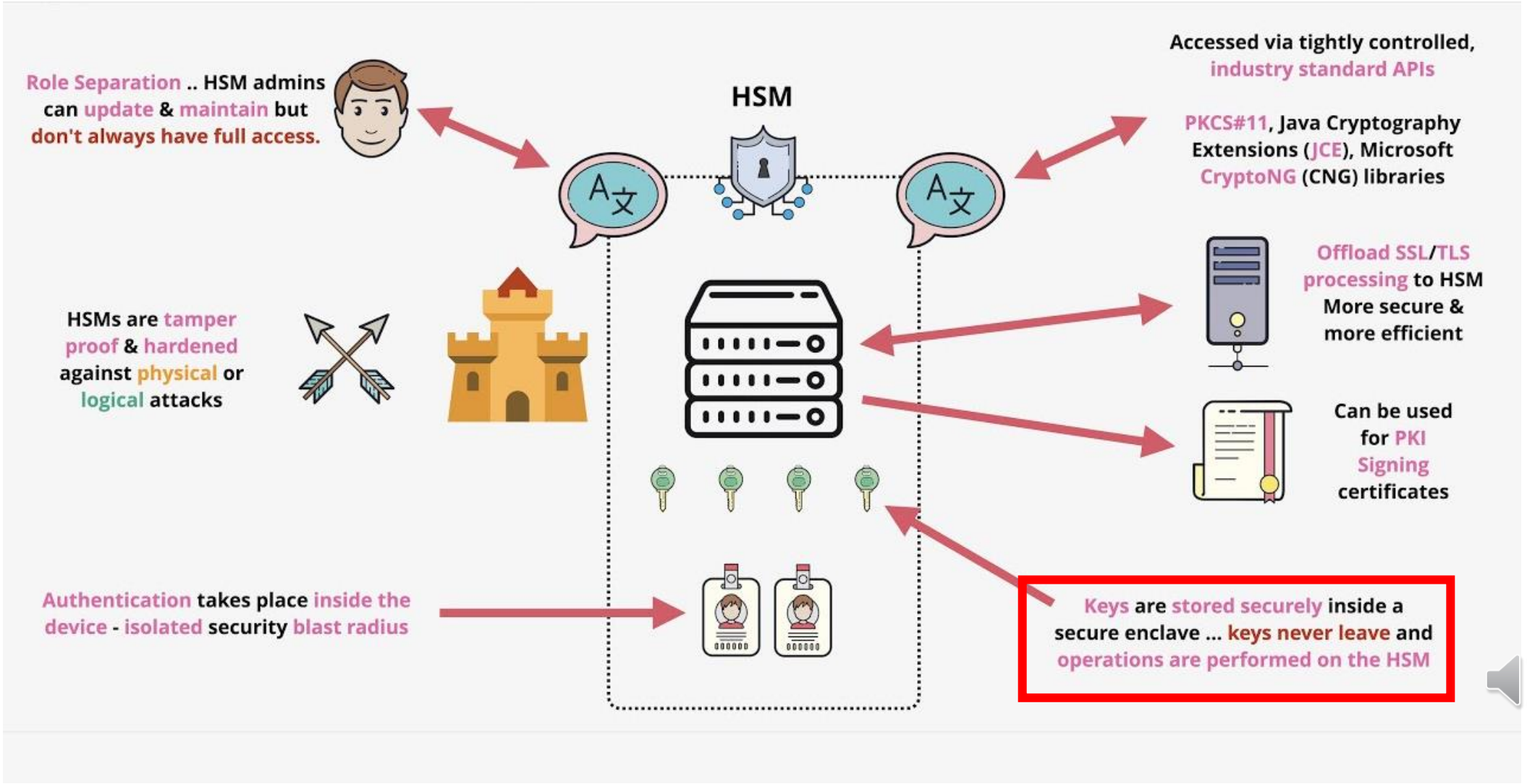
Hardware Security Module (HSM) [2]



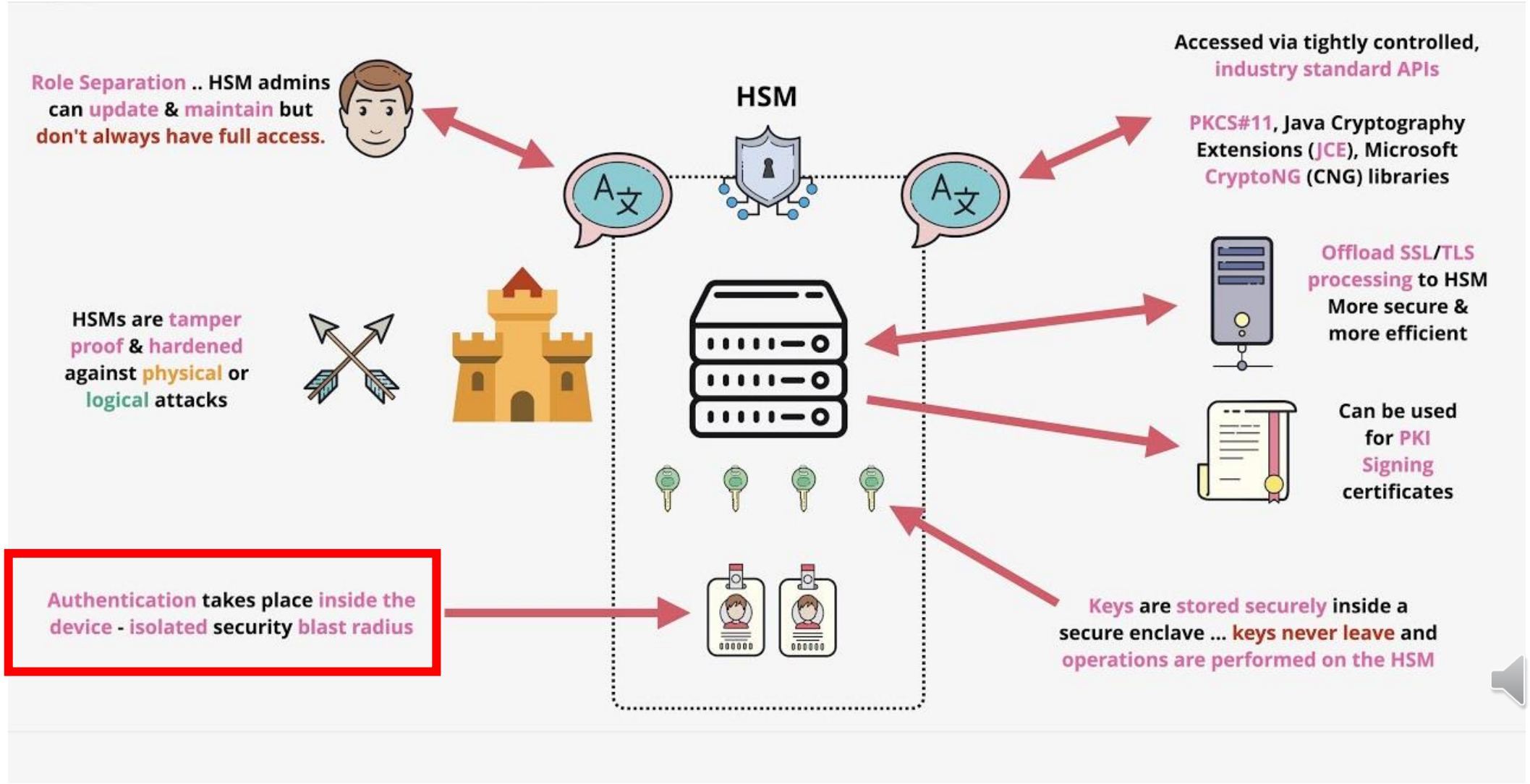
Hardware Security Module (HSM) [2]



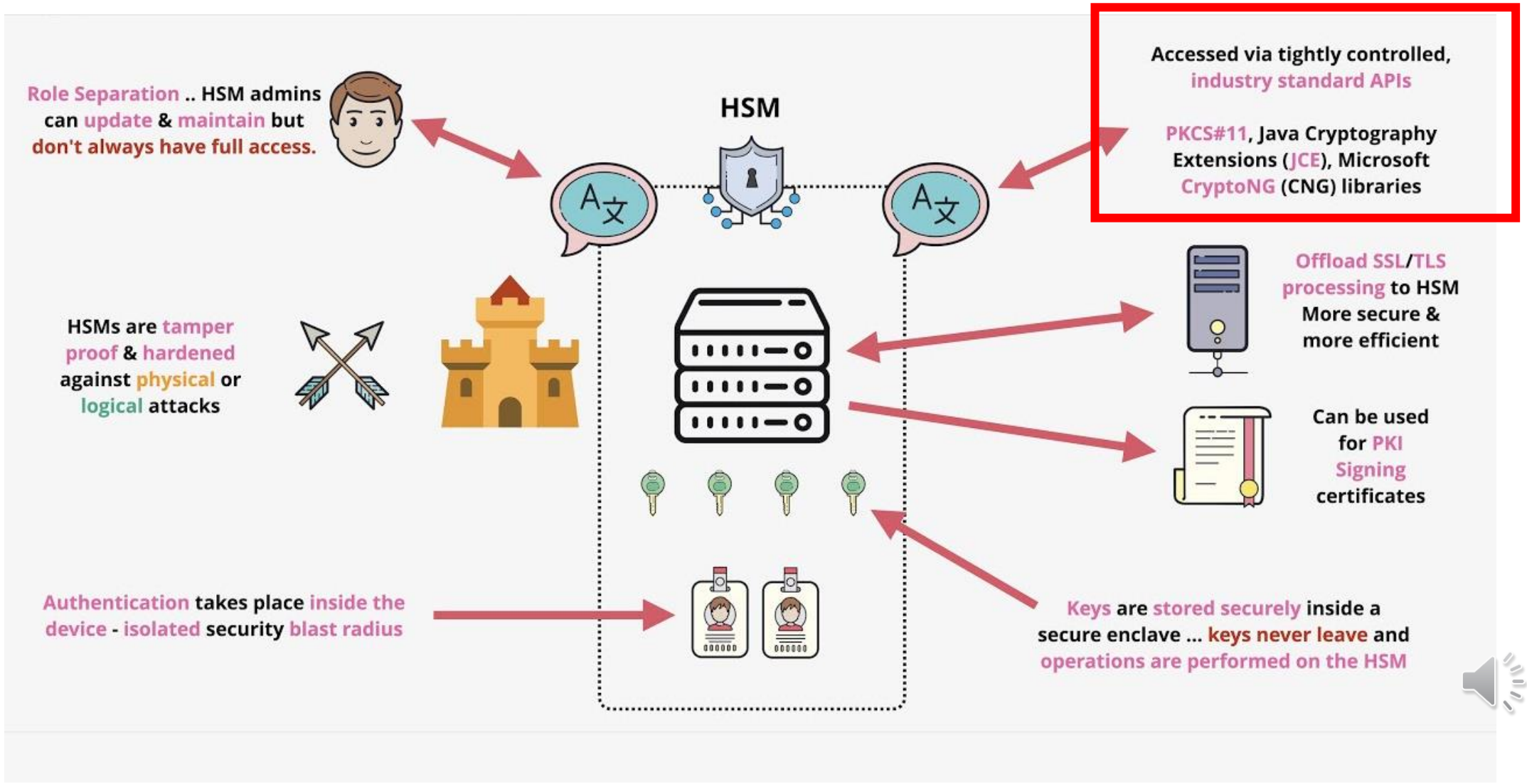
Hardware Security Module (HSM) [2]



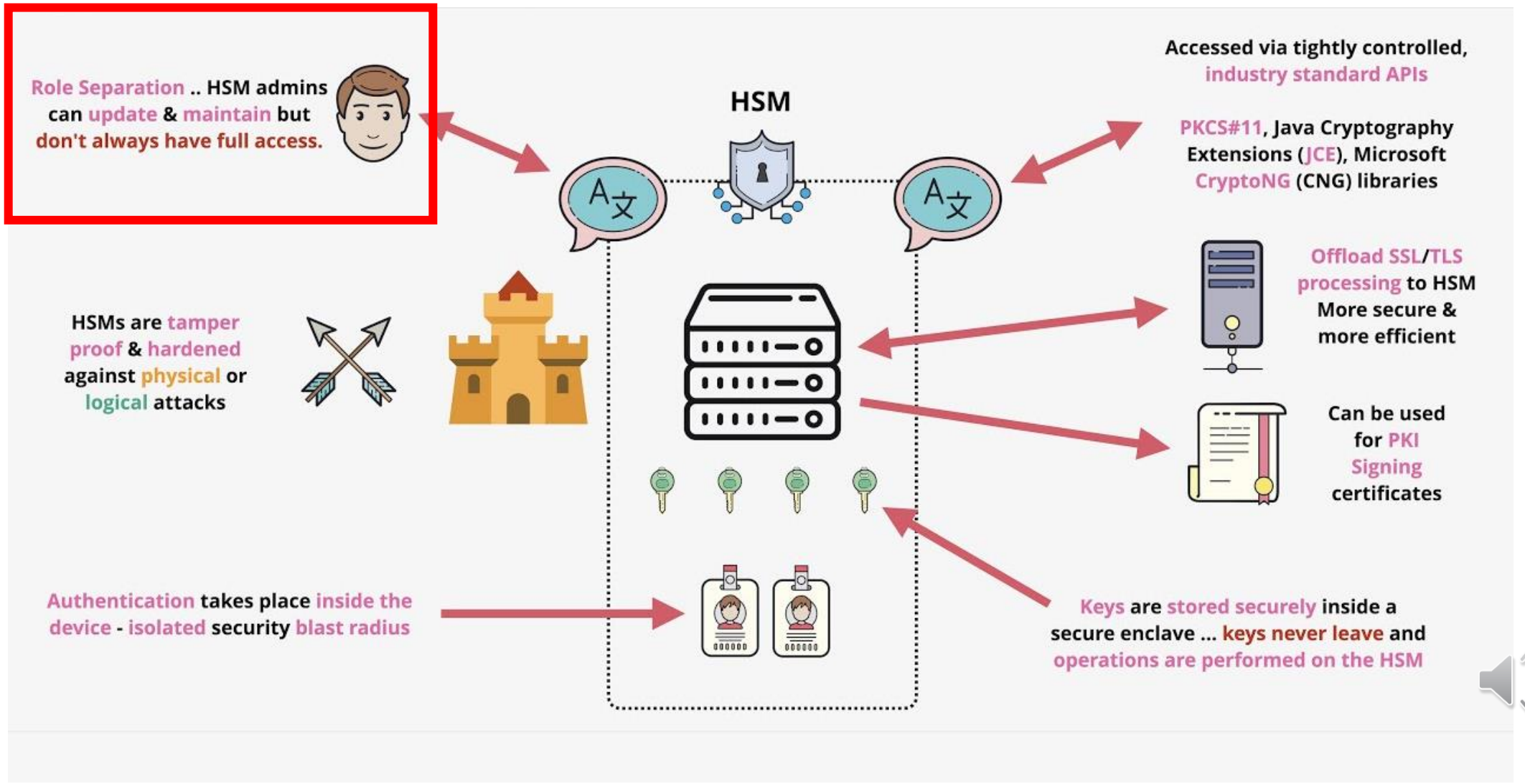
Hardware Security Module (HSM) [2]



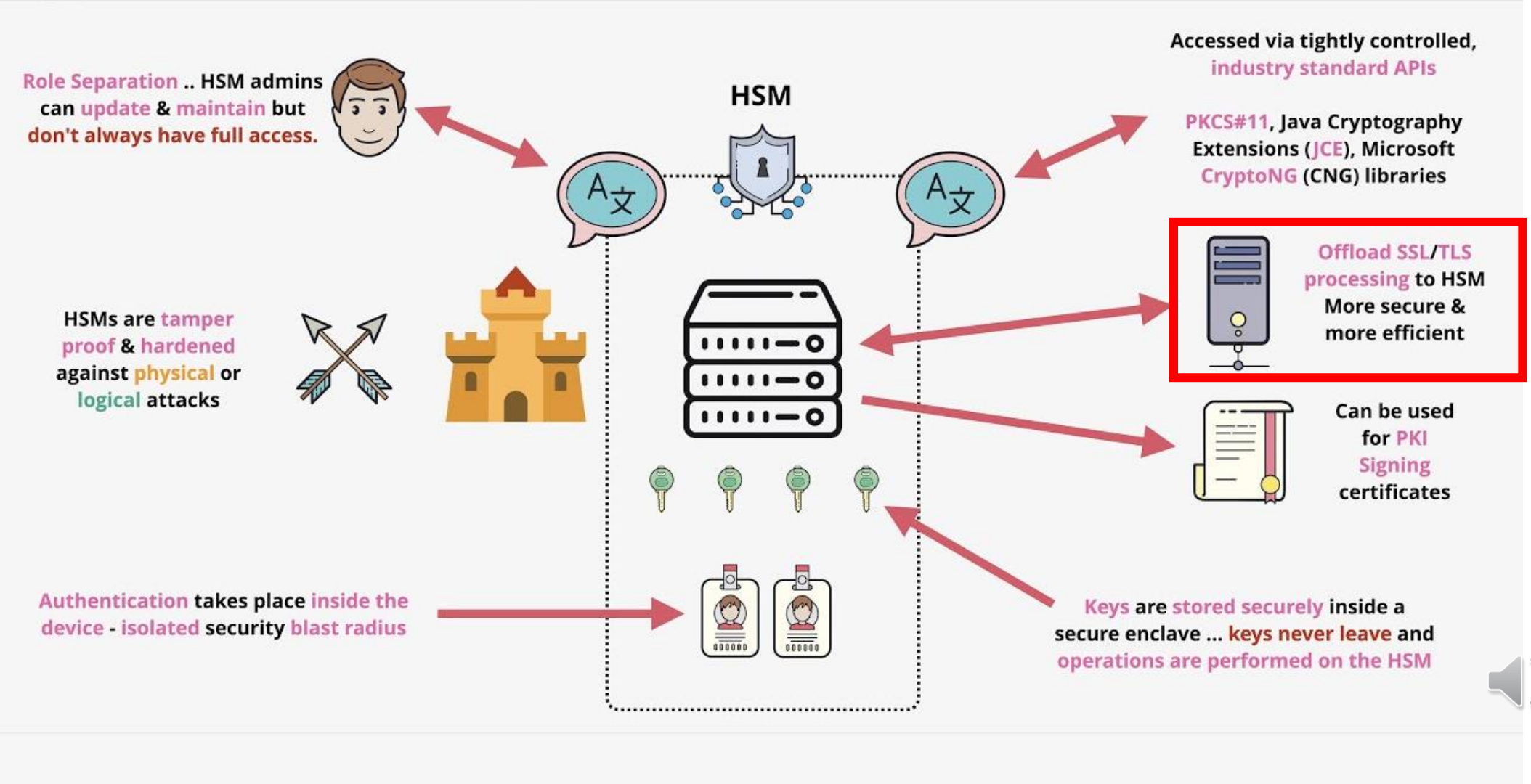
Hardware Security Module (HSM) [2]



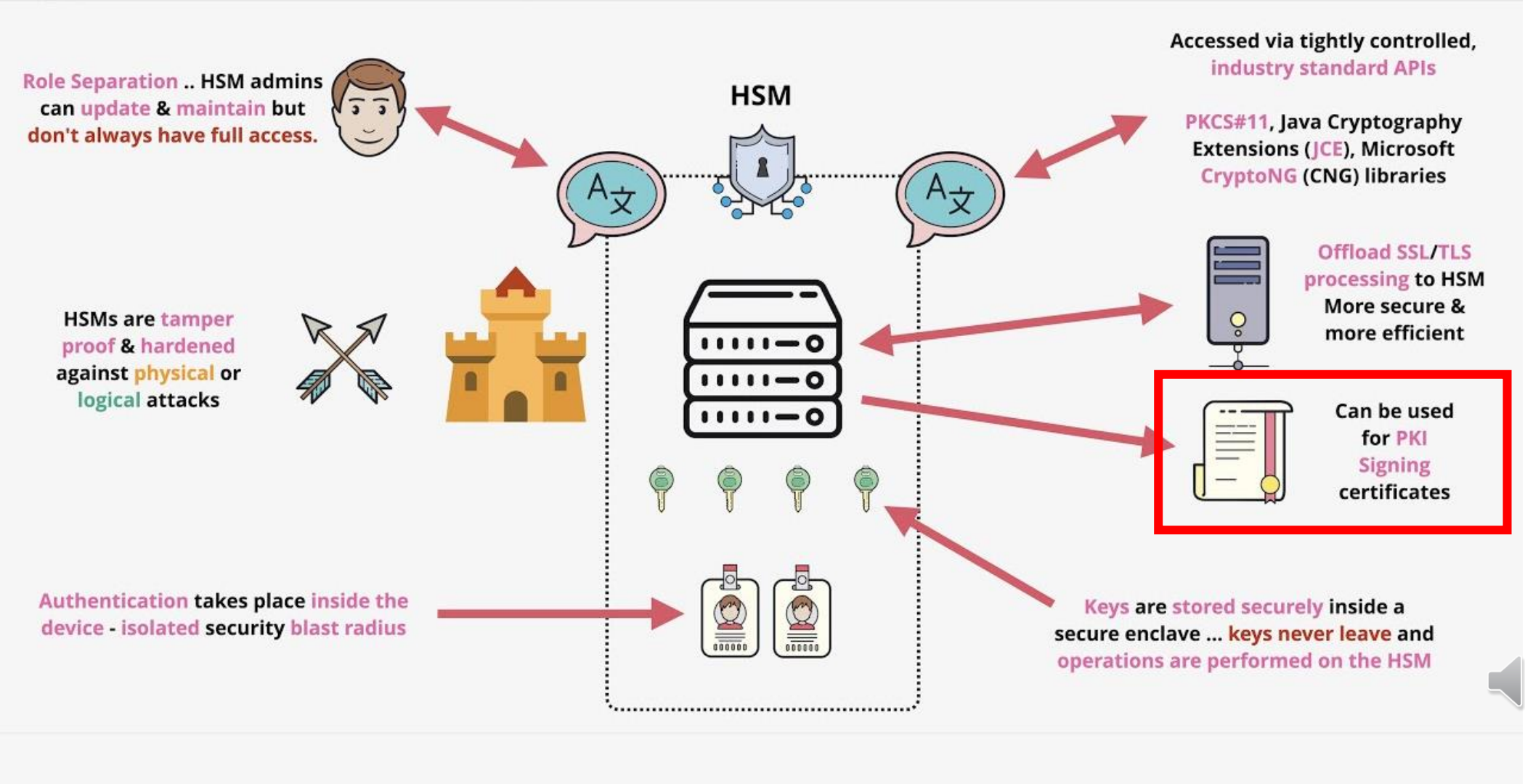
Hardware Security Module (HSM) [2]



Hardware Security Module (HSM) [2]



Hardware Security Module (HSM) [2]



Hardware Security Module (HSM)

- Definition:
  - HSMs are **specialized** devices used to **conduct** cryptographic operations and use a random number source to **generate** public-private key pairs and subsequently **store** them [1]
- Isolated cryptographic processes from other operations
  - more efficient processing and additional security [4]
- Comes in various form factors [3]
  - Embedded PCI Express card
  - Standalone Ethernet connected appliances
  - USB connected devices
- Use cases
  - Protection of privileged access and company secrets
  - Keys Generation, Storage, and Management
  - Authentication and identity management
  - Compliance and Auditing
- Standard
  - NIST FIPS - 140-2 security level 3 [4]

Hardware Security Module (HSM)

- Features [5]
  - Secure storage
    - Stores keys in encrypted form
    - Multi-level encryption
  - Performance and high availability
    - High speed cryptography operations
    - No need to run in the background
    - Secure backups and recovery
  - Tamper-resistance
    - Prevent intrusion
  - Tamper-evidence
    - Leaving special seeds or stickers evidence when using HSM
- Single HSM or HSM clusters

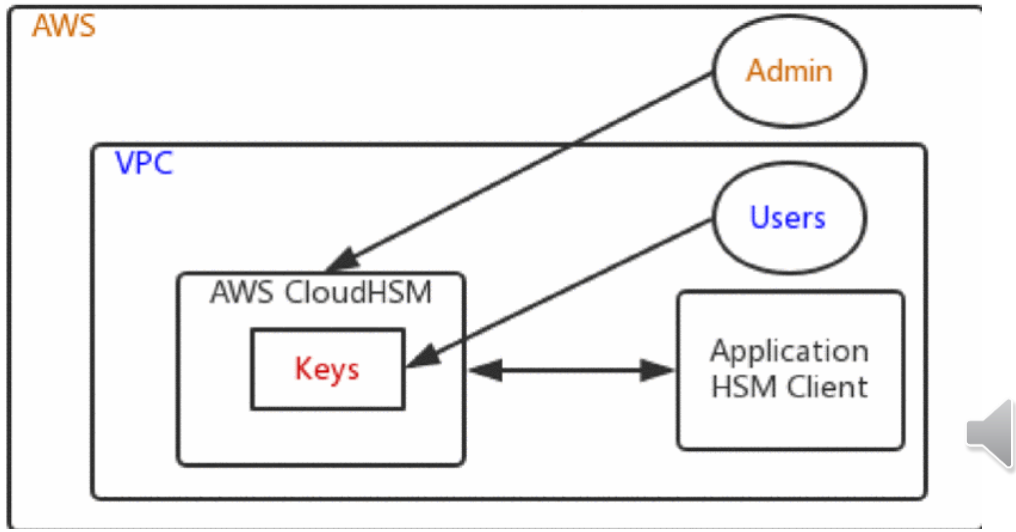
Tenant Separation [6]

- Partitioning
- Access control
- Key separation
- Logging and auditing

- Dedicated Data Security at Large Scale
  - To manage the keys, data and security policies of each tenant
- Proper Implementation of Authentication Mechanisms
  - Each HSM is single-resident-at-a-time
- Easy Integration and Management
- Less Deployment & Integration Costs
  - Procurement, deployment, and management
- Threat Detection and Audit Management
  - Advanced level logging, alerting and audit mechanisms

Cloud HSM [7]

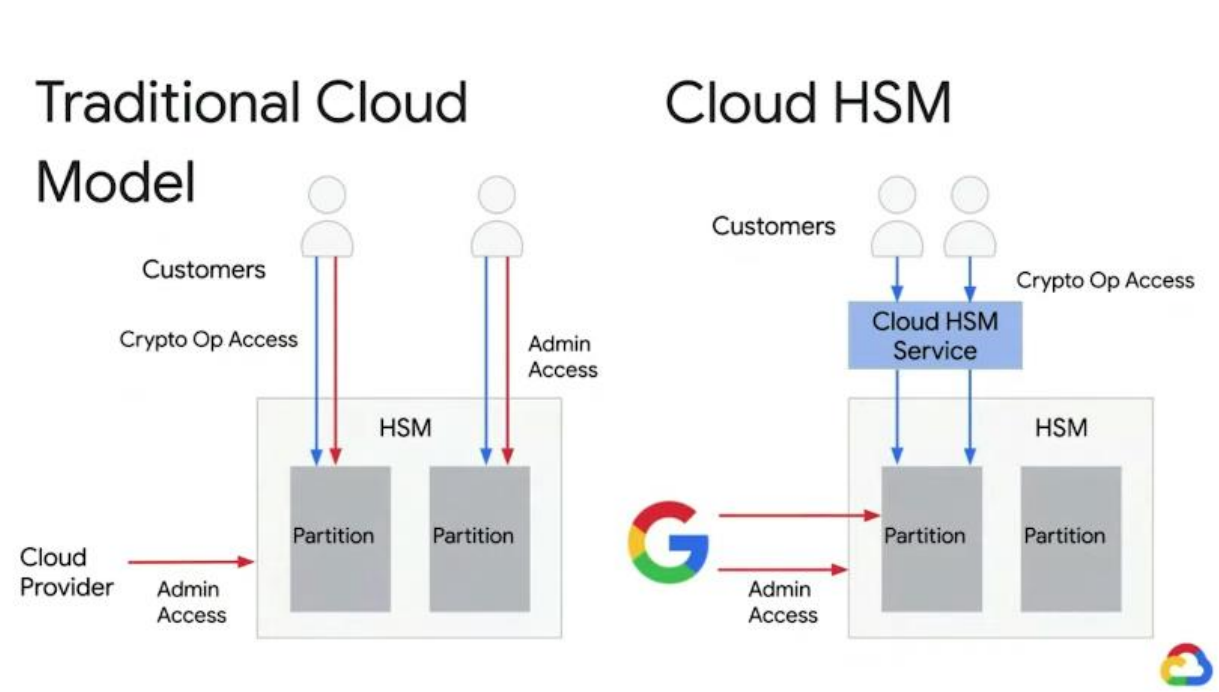
- To better utilize an HSM device, businesses consider the adoption of cloud-based HSMs
- A cloud-based HSM is still a physical device but is kept in a cloud data center
  - HSMaaS
- Operations:
  - Generate, store, import, export, and manage cryptographic keys.
  - Encrypt and decrypt data with symmetric or asymmetric algorithms.
  - Compute message digests and hash-based message authentication codes by cryptographic hash functions.
  - Sign data cryptographically and verify signatures.
  - Generate secure random data cryptographically.
- Several cloud service providers give the possibility to use real HSM clusters managed entirely in the cloud and accessible via remote APIs [9]
  - AWS, IBM, Microsoft Azure, Google, …etc.

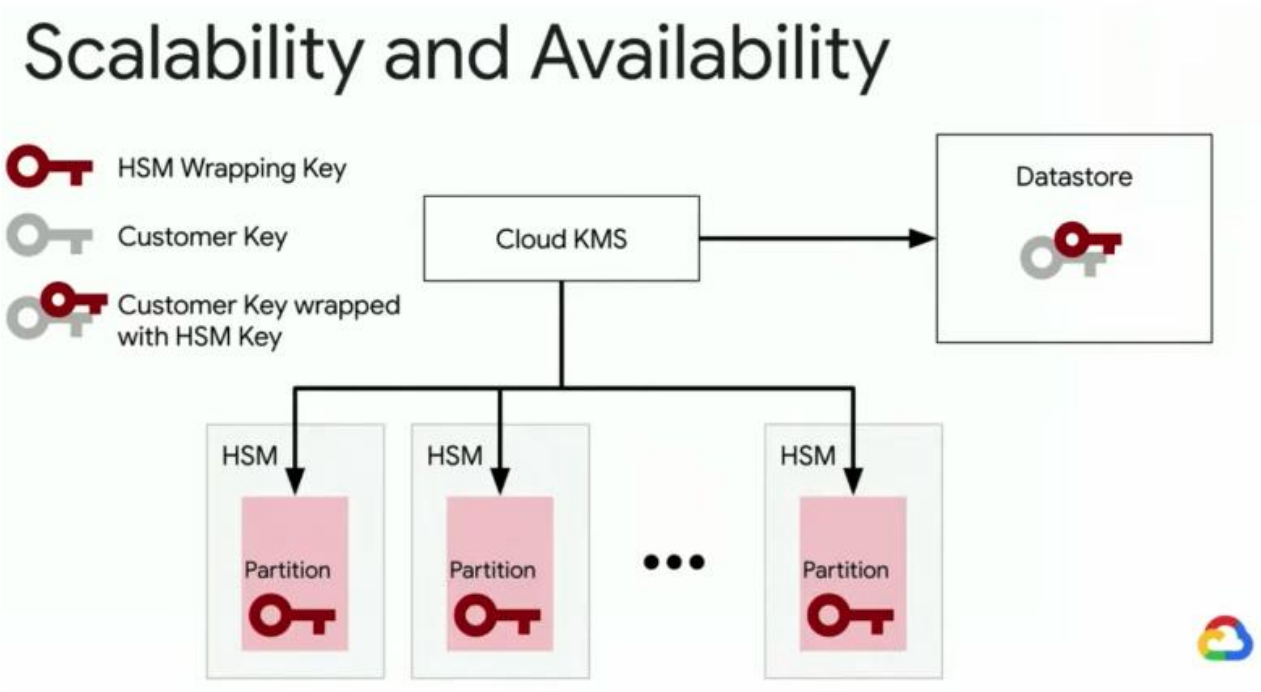


VPC: Virtual Private Cloud

Cloud HSM [8]

Google Cloud



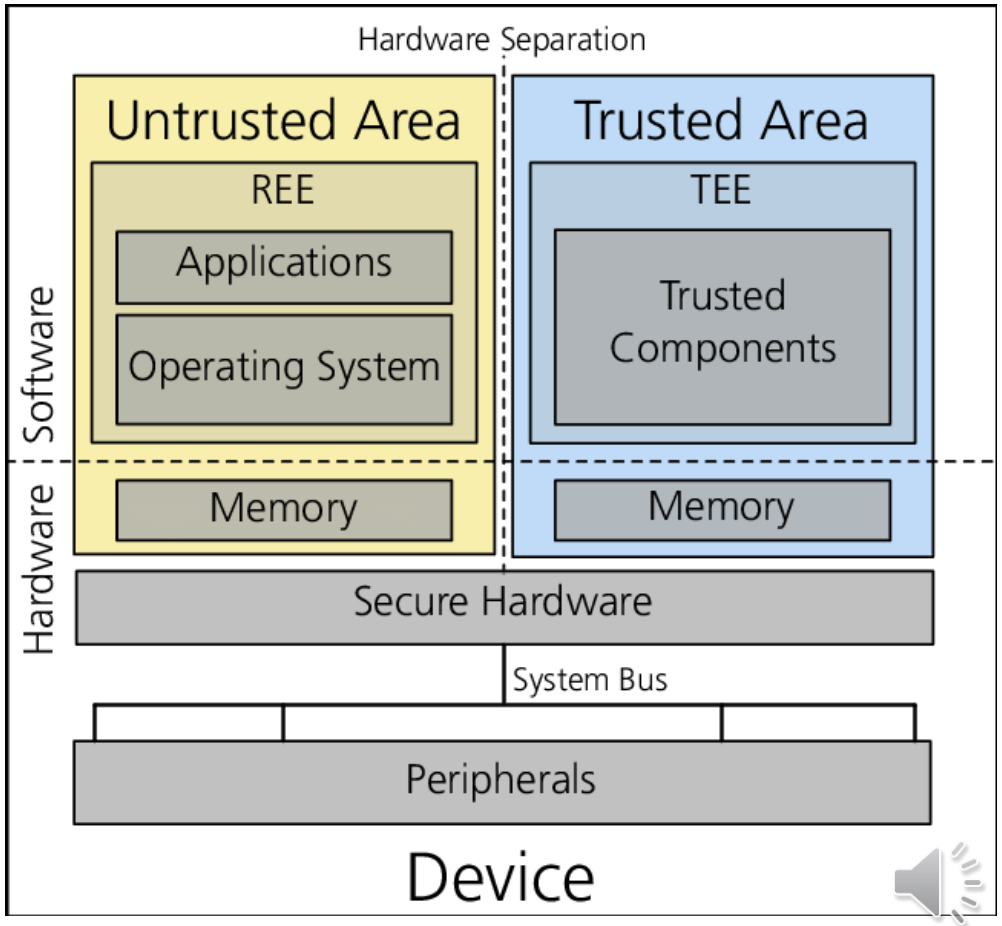


Related Complementary: TPM v.s. HSM

- TPM: Trusted Platform Module
  - Small hardware devices that are usually embedded into computer
  - TPM contains a secret key which is not accessible to the outside world
- HSM: Hardware Security Module
  - Either embedded or standalone device
  - Allow applications to easily generate secrets and perform operations on secrets, but they do not allow them to easily read secrets
- Comparisons
  - HSMs are more powerful
    - High end HSMs can be faster than a computer CPU when performing cryptographic operations
    - Provide meaningful acceleration of encryption or decryption
  - HSMs are generic devices that conform to APIs such as PKCS #11
    - They are accessible to any application that wants to use their services
    - While TPMs are usually more closely integrated with their host computers, their operating system, their booting sequence, or the built-in hard drive encryption
  - HSMs are meant to be used in data centers, while the scope of a TPM is usually a single system

Related Complementary: TEE (Trusted Execution Environment)

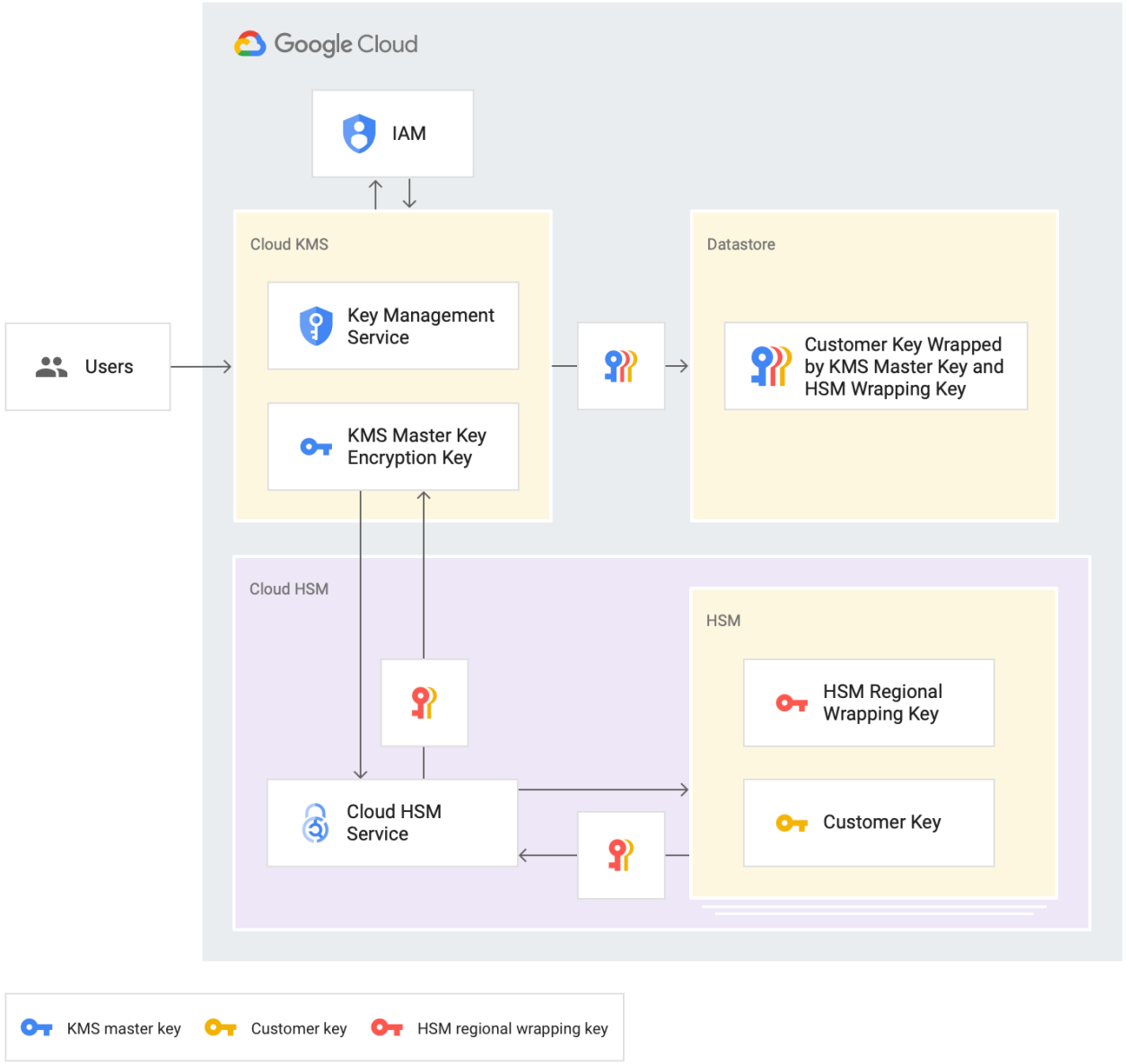
- A **secure** and **isolated** environment within a processor, ensuring the confidentiality and integrity of sensitive data and code execution
- Secure Isolation:
  - TEE operates in a separate execution environment, isolated from the main operating system and other applications.
- Trusted Execution:
  - Code and data executed within the TEE are verified and protected from tampering or unauthorized access.
- Secure Communication:
  - TEE enables secure communication channels between trusted applications and peripherals, ensuring data confidentiality and integrity.
- Hardware-backed Security:
  - TEE leverages hardware-based security features, such as secure enclaves and cryptographic accelerators, to enhance protection against attacks.



Related Complementary: Cloud KMS v.s. Cloud HSM

- Cloud Key Management Service (KMS):
  - A managed service offered by cloud providers for generating, storing, and managing encryption keys
  - Provides centralized key management with integrated rotation and auditing capabilities
  - Suitable for applications requiring flexible and scalable key management solutions
- Cloud Hardware Security Module (HSM):
  - A dedicated hardware device used to generate, store, and manage cryptographic keys
- Security Features:
  - Cloud KMS: **Software-based** security with encryption at rest and in transit.
  - Cloud HSM: **Hardware-based** security with physical protection against tampering and key extraction.
- Performance:
  - Cloud KMS: Offers **high scalability and availability** but may have variable performance.
  - Cloud HSM: Provides **consistent performance** with dedicated hardware acceleration.
- Cost and Management:
  - Cloud KMS: Typically, more cost-effective and **easier to manage** due to its fully managed nature.
  - Cloud HSM: **Higher upfront costs** and requires specialized expertise for management and maintenance.
- Compliance and Assurance:
  - Cloud KMS: **Compliance certifications** but may not meet all regulatory requirements.
  - Cloud HSM: Often chosen for compliance-sensitive environments due to its stringent security measures.

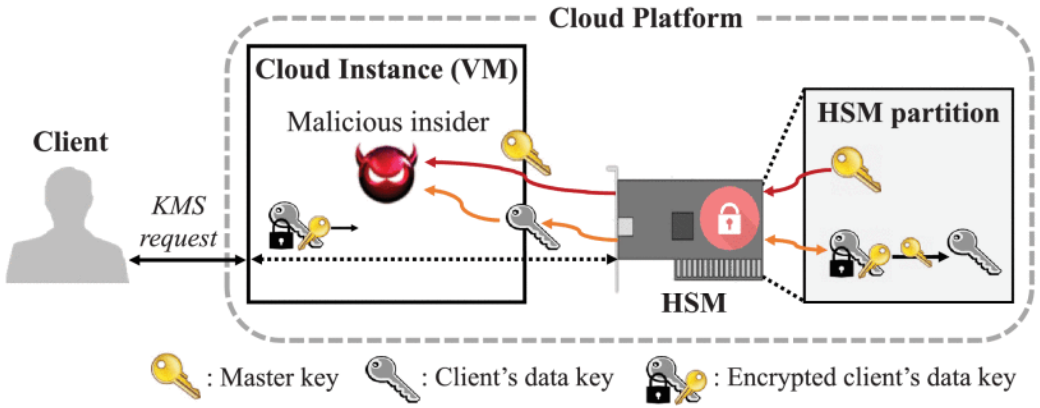
Related Complementary: Cloud KMS v.s. Cloud HSM



Scalable and Secure Virtualization of HSM With ScaleTrust [10]

Problem addressed

- Insider threat in cloud KMS
  - No protection against a malicious insider (i.e., a cloud provider) who has control over the HSM



This environment allows the KMS to fully control communications between the client and the HSM

1. Freely eavesdrop on their traffic
2. Send arbitrary commands to the HSM
3. Leaking or misusing a client's secret key
4. Go undetected because the insider can modify the KMS log

- Limitations of existing approaches
  - Replaces the KMS with SGX, establishing a trusted computing environment that offers confidentiality and integrity
  - But SGX suffers from various types of side-channel attacks
  - Intel® Software Guard Extensions (SGX) : a set of instruction codes implementing trusted execution environment (TEE), to allow user-level and operating system code to define protected private regions of memory, called **enclaves**
  - Side-channel attacks:
    - Implementation specific attack
    - Exploit that variations in algorithm behavior resulting from different inputs to infer secret values like cryptographic keys, ex: computation time

Proposed method: ScaleTrust

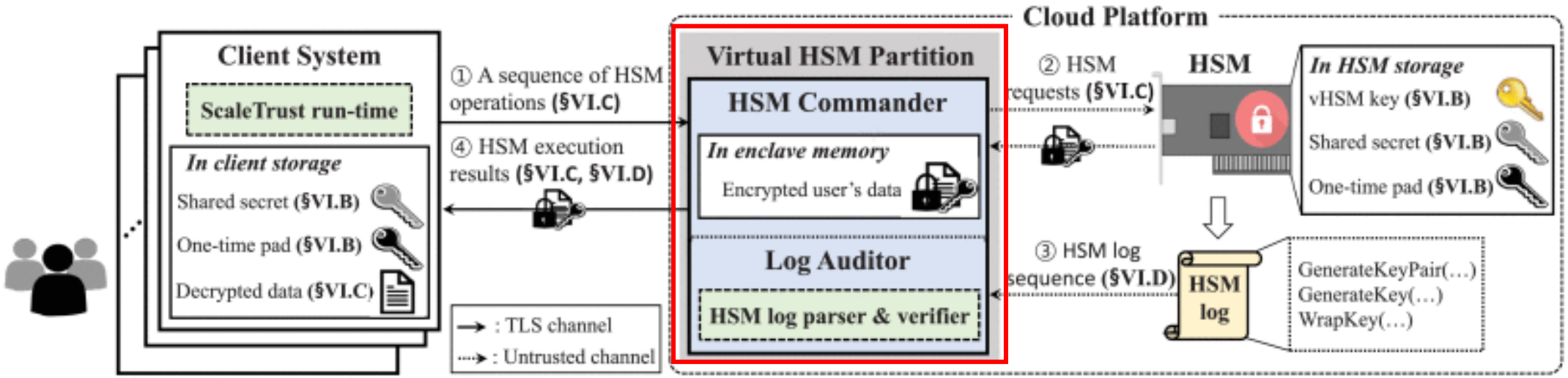
- Design Goals;
  - G1: Protection against insider threats
  - G2: Key usage isolation for multi-tenancy
  - G3: FIPS-grade protection for the root of trust
  - G4: SGX side-channel attack mitigation

TEE: A secure area of a main processor. It helps code and data loaded inside it to be protected with respect to confidentiality and integrity.

- Assumptions
  - A powerful adversary attempts to **directly** obtain or abuse cryptographic keys stored in an HSM or CPU-hardened TEE (Trusted Execution Environment)
  - The adversary shares the **same** host or cloud platform with victims.
  - The adversary has **full control** over the system software, such as the operating system and the hypervisor, thus able to **forge** the communication between the users and HSMs by manipulating the system memory
  - An attacker can even **leak** HSM-generated keys in enclaves using side-channel attacks
  - Adversary is not powerful enough to **break an enclave's integrity** nor capable of compromising any confidentiality requirements
  - **HSMs are trustworthy**
  - A PKC (Public-Key Certificate) certificate from an HSM is trustworthy
  - Trust the SGX enclave's code integrity and verification of (SGX-enabled) the platform's genuineness

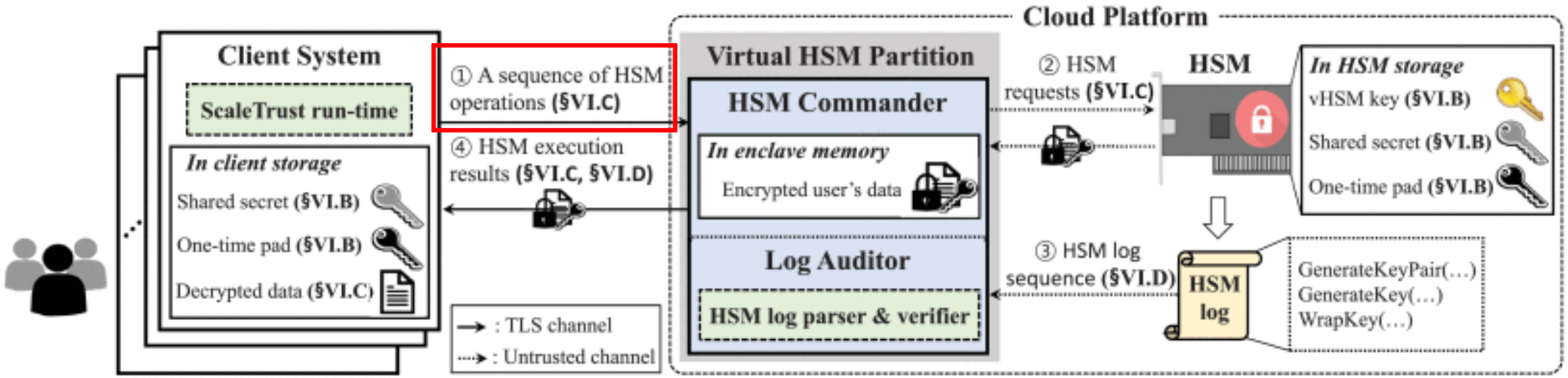
ScaleTrust

- HSM Commander Thread (Commander)
  - a secure bridge between an HSM and a KMS client, isolating in-HSM key usages for each client
- Log Auditor Thread (Auditor)
  - detecting any unauthorized access to in-HSM keys by verifying the HSM log



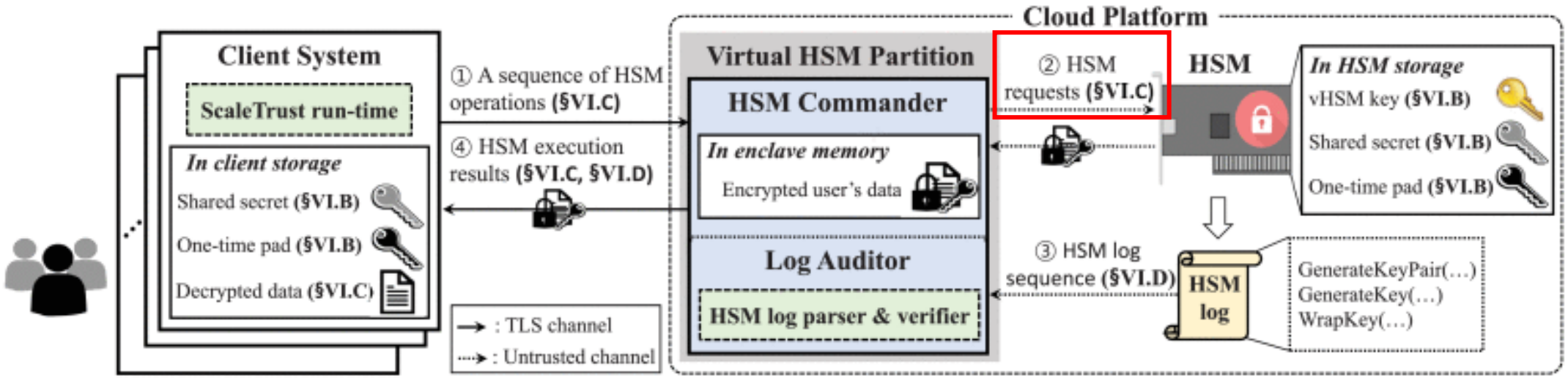
ScaleTrust

- HSM Commander Thread (Commander)
  - a secure bridge between an HSM and a KMS client, isolating in-HSM key usages for each client
- Log Auditor Thread (Auditor)
  - detecting any unauthorized access to in-HSM keys by verifying the HSM log



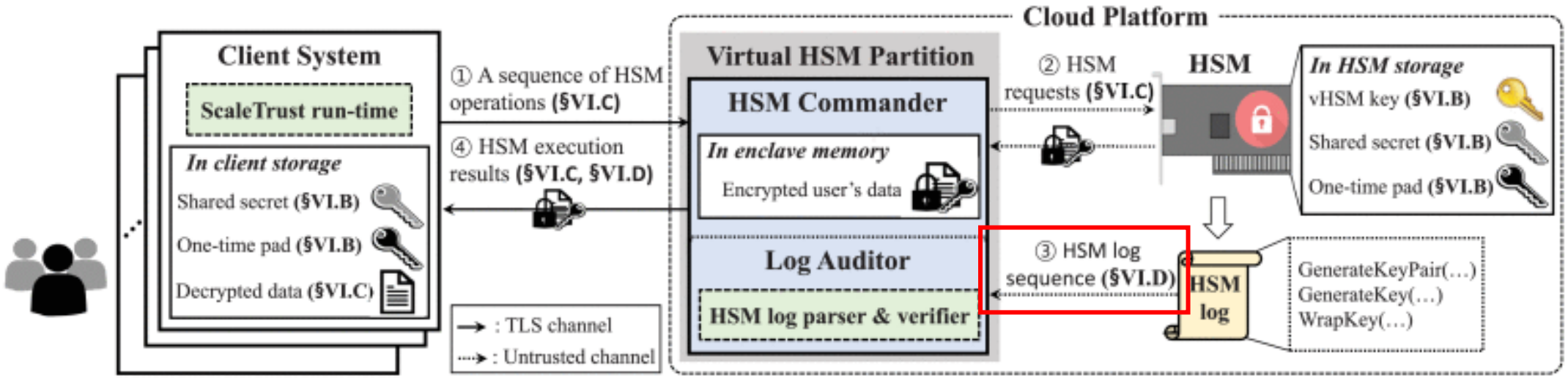
ScaleTrust

- HSM Commander Thread (Commander)
  - a secure bridge between an HSM and a KMS client, isolating in-HSM key usages for each client
- Log Auditor Thread (Auditor)
  - detecting any unauthorized access to in-HSM keys by verifying the HSM log



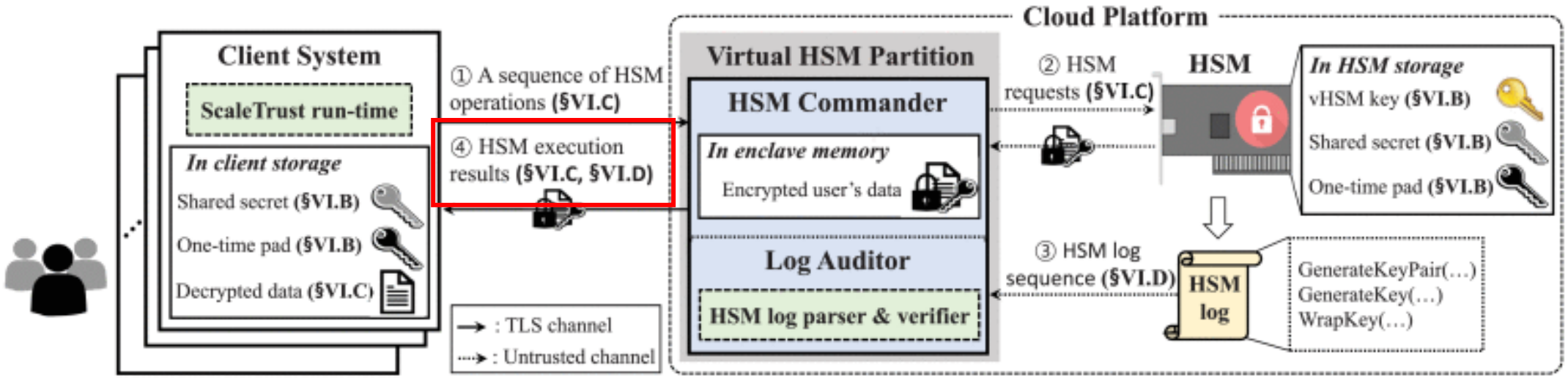
ScaleTrust

- HSM Commander Thread (Commander)
  - a secure bridge between an HSM and a KMS client, isolating in-HSM key usages for each client
- Log Auditor Thread (Auditor)
  - detecting any unauthorized access to in-HSM keys by verifying the HSM log

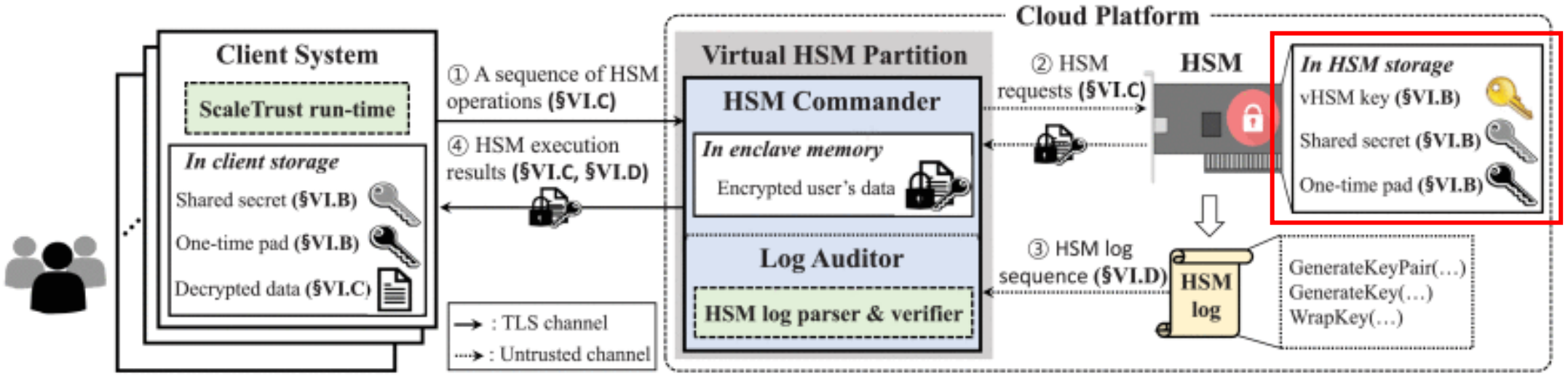


ScaleTrust

- HSM Commander Thread (Commander)
  - a secure bridge between an HSM and a KMS client, isolating in-HSM key usages for each client
- Log Auditor Thread (Auditor)
  - detecting any unauthorized access to in-HSM keys by verifying the HSM log



ScaleTrust



- vHSM creation:
  - Generating a non-extractable RSA key pair stored within the HSM, ensuring the private key remains within the device and serves as the root of trust for subsequent keys derived from it.
- Establishing shared secret:
  - For secure communication, encrypting and decrypting messages using public and private keys respectively, preventing unauthorized access by insiders.
- One-time pad:
  - Encrypted using the shared secret, ensuring secure channel establishment between the client and the HSM, mitigating potential decryption threats.

Security analysis

- ScaleTrust is secure under attacks against HSMs and the channel between clients and enclaves
- Attacks on Enclaves
  - ScaleTrust is secure under attacks that try to achieve client master keys stored in the cloud platform
- Attacks on Enclaves ↔ HSMs
  - Eavesdropping
  - Manipulating data messages
  - Manipulating HSM control messages
- Worst-case security impact
  - manipulate HSM commands (only detection)
    - The vulnerability of in-use keys
- Attacks on client
  - ScaleTrust decrypts and uses the encrypted client master keys **only** in the HSM and never exports them to client

Threats on KMS	ScaleTrust	HSM	TEE	HSM-TEE
Attacks on Enclaves				
Side-channel attacks	M	N/A	✗	✗
Attacks on Cloud Instances (e.g., Enclaves) ↔ HSMs				
Eavesdropping	P	✗	N/A	P
Manipulating data messages	P	✗	N/A	P
Manipulating HSM commands	D	✗	N/A	✗
Attacks on Clients				
Client master key leakage	P	P	P	P

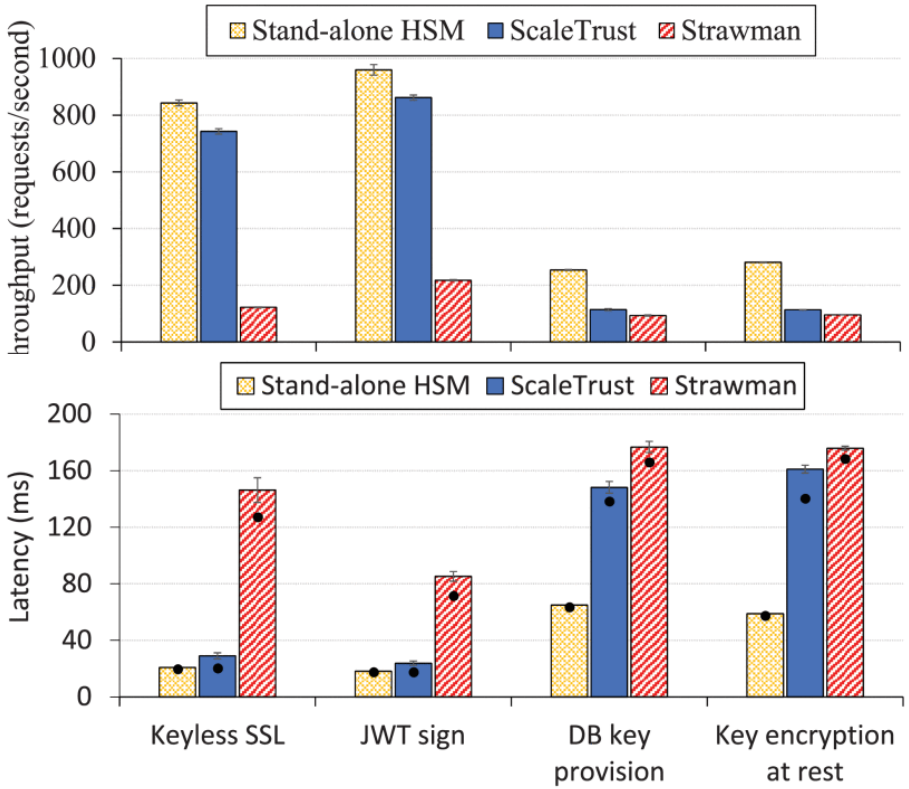
HSM: HSM-backed Cloud KMS, **TEE**: TEE-only Cloud KMS,
HSM-TEE: HSM-TEE hybrid KMS,
P: Prevention, **M**: Mitigation, **D**: Detection.

Evaluation

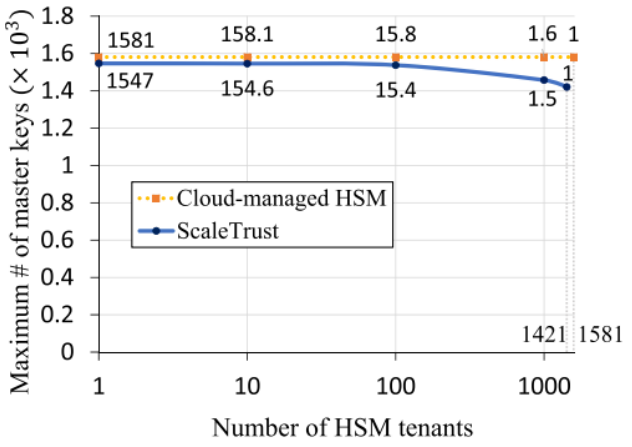
- Performance compared to a stand-alone HSM

Operation	Stand-alone HSM # of commands				ScaleTrust # of commands			
	G	E	S	O	G	E	S	O
Master key signing	0	0	1	0	1*	3*	1	4*
Key provisioning	2	2	0	1	3	3	0	5
Key encryption at rest	1	2	0	0	2	4	0	5

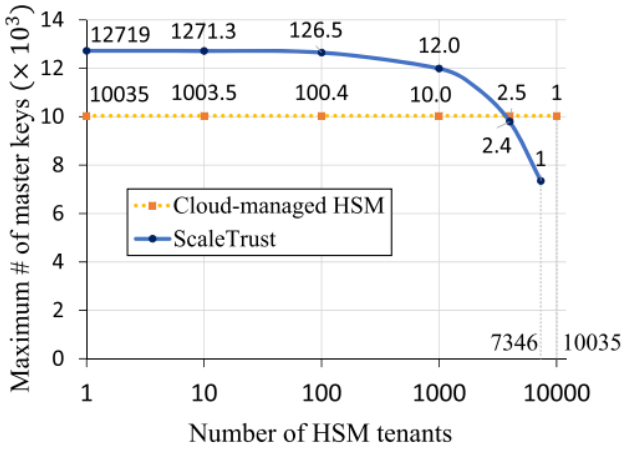
G: Generate key (pair), E: Encrypt, decrypt, wrap, and unwrap, S: Sign, O: Other HSM commands (e.g., Destroy key), *: occurs only during master key restore.



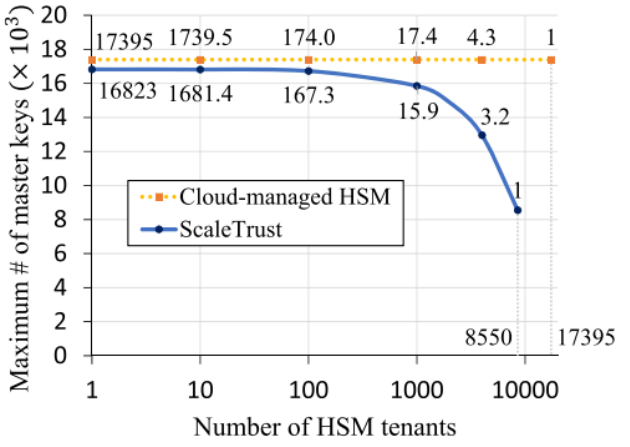
- Multi-tenancy



(a) 2048-bit RSA key for Keyless SSL.



(b) P-256 EC key for JWT signing service.

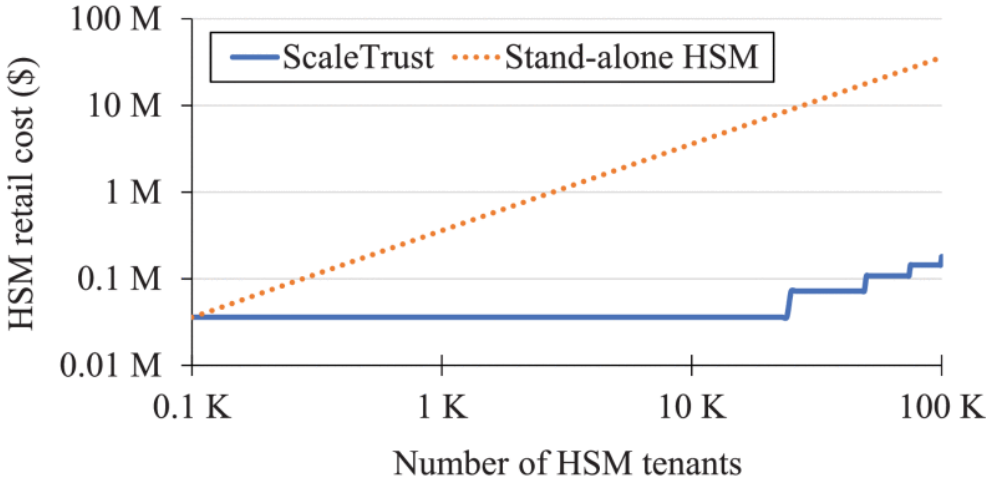


(c) 256-bit AES key for Key encryption at rest service.

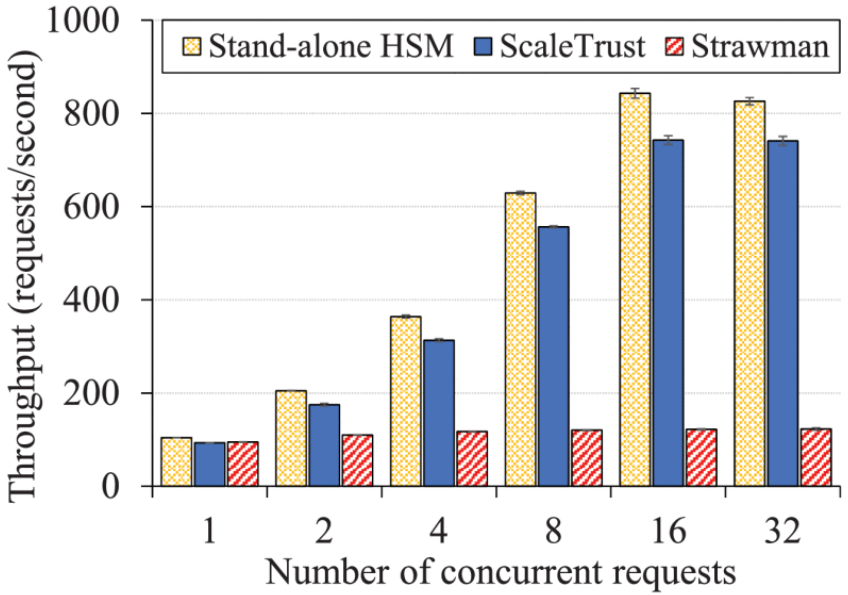
Evaluation

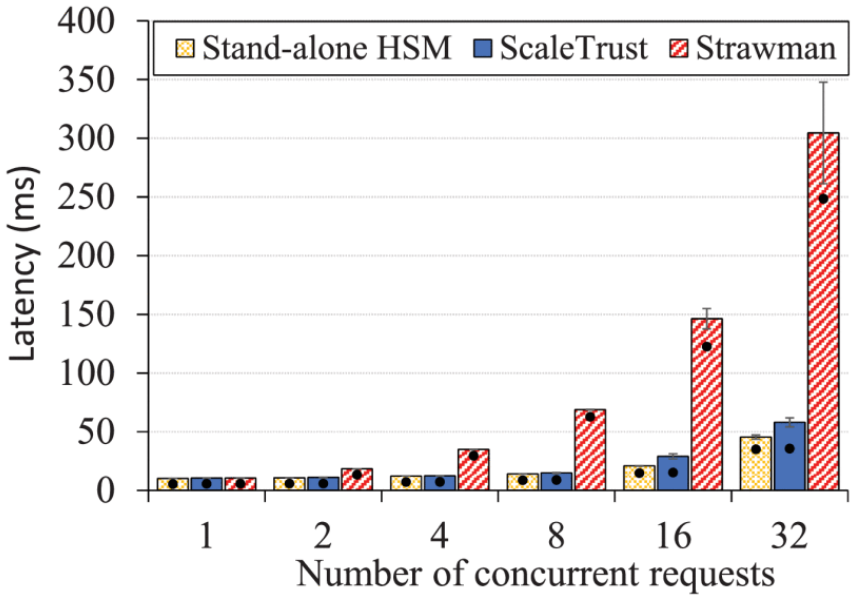
- estimated HSM cost
  - each tenant uses a 2048-bit RSA private key within an HSM

Key stored in HSM	Key size (B)	Metadata size (B)	Total size (B)
Plain-text RSA key	1,224	96	1,320
Plain-text EC key	72	136	208
Plain-text AES key	32	88	120
Encrypted RSA key	1,232	116	1,348
Encrypted EC key	80	84	164
Encrypted AES key	40	84	124



- Concurrent execution





Conclusion

- HSM
  - Specialized devices used to conduct cryptographic operations
- cloudHSM
  - Cloud services, HSM-as-a-service
  - Use it when need it
- Related Complementary
  - Trusted Platform Module, Cloud Key Management Service, Trusted Execution Environment
- ScaleTrust
  - Multi-tenant isolation
    - ScaleTrust ensures secure isolation of legacy HSMs in cloud environments, even against malicious insiders.
  - PKCS #11 support
    - ScaleTrust offers comprehensive support for PKCS #11 operations, safeguarding against insider threats by translating cryptographic requests into secure HSM commands.
  - Scalable and practical
    - ScaleTrust's evaluation demonstrates practicality and scalability, supporting real-world applications with performance comparable to stand-alone HSMs.

Insights and Future works

- Traditional PKC, stores keys in HSM and the memory (in TEE, enclaves) when using
  - PUF-based HSM and PUF-based cloud HSM
  - How it will be implemented and what is the difference between current HSM?
  - **Key** size: 256 bits~ 2048 bits /tenant v.s. **Challenge-Response Pair**: 64+1 bits~256+1bits /tenant

Reference

1. Sommerhalder, Maria. "Hardware Security Module." *Trends in Data Protection and Encryption Technologies* (2023): 83-87.
2. What are hardware security modules (HSM), why we need them and how they work. (https://youtu.be/szagwwSLbXo?si=MreKZk912)
3. National Institute of Standards and Technology. Security Requirements for Cryptographic Modules. Technical Report Federal Information Processing Standard (FIPS) 140-2, U.S. Department of Commerce, December 2002, pp.28-30
4. What is a hardware security module (HSM)? (https://youtu.be/9BEqlZjQNio?si=DSsuiwei1hKayOOpLkMPAYW)
5. Cryptography : What are Hardware Security Modules (HSM)? (https://youtu.be/R25z38iP3m4?si=Ni9wac0MUVEz5DKx)
6. Utimaco. 2021. Cloud HSM. Accessed in May 2021 at https://hsm.utimaco.com/products-hardware-security-modules/form-factor/cloud-hsm-as-a-service/ .
7. X. Huang and R. Chen, "A Survey of Key Management Service in Cloud," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 916-919, doi: 10.1109/ICSESS.2018.8663805.
8. Cloud HSM deep dive and best practices (https://youtu.be/mm-y-TKWjgg?si=pmZlvEx1YiikCJxn)
9. Focardi, Riccardo, and Flaminia L. Luccio. "A formally verified configuration for hardware security modules in the cloud." Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021.
10. Han, Juhyeng, et al. "Scalable and Secure Virtualization of HSM With ScaleTrust." IEEE/ACM Transactions on Networking (2022).
11. Wang, Yongge. "Public key cryptography standards: PKCS." *arXiv preprint arXiv:1207.5446* (2012).
12. IBM Cloud Hardware Security Module (HSM) (https://www.ibm.com/docs/en/blockchain-platform/2.5.1?topic=tutorials-hardware-security-module-hsm)

Appendix 1: NIPS 140-2 security level 3

In addition to the applicable requirements for Security Levels 1 and 2, the following requirements shall apply for Security Level 3.

- All cryptographic software and firmware, cryptographic keys and CSPs, and control and status information shall be under the control of
  - an operating system that meets the functional requirements specified in the Protection Profiles listed in Annex B. The operating system shall be evaluated at the CC evaluation assurance level EAL3 and include the following additional requirements: Trusted Path (FTP_TRP.1) and Informal TOE Security Policy Model (ADV_SPM.1), or
  - an equivalent evaluated trusted operating system.
- All cryptographic keys and CSPs, authentication data, control inputs, and status outputs shall be communicated via a trusted mechanism (e.g., a dedicated I/O physical port or a trusted path). If a trusted path is used, the Target of Evaluation Security Functions (TSF) shall support the trusted path between the TSF and the operator when a positive TSF-to-operator connection is required. Communications via this trusted path shall be activated exclusively by an operator or the TSF and shall be logically isolated from other paths.
- In addition to the audit requirements of Security Level 2, the following events shall be recorded by the audit mechanism:
  - attempts to use the trusted path function, and
  - identification of the initiator and target of a trusted path.

Appendix 2: PKCS#11

- PKCS #11 (Public-Key Cryptography Standards #11) is a widely used cryptographic API (Application Programming Interface) standard that defines a platform-independent interface for accessing cryptographic tokens such as hardware security modules (HSMs) and smart cards.
- Developed by RSA Laboratories, PKCS #11 provides a uniform interface for cryptographic operations, key management, and secure storage across different cryptographic devices and software applications.

Table 30, Summary of Cryptoki Functions

Category	Function	Description
General purpose functions	C_Initialize	initializes Cryptoki
	C_Finalize	clean up miscellaneous Cryptoki-associated resources
	C_GetInfo	obtains general information about Cryptoki
	C_GetFunctionList	obtains entry points of Cryptoki library functions
Slot and token management functions	C_GetSlotList	obtains a list of slots in the system
	C_GetSlotInfo	obtains information about a particular slot
	C_GetTokenInfo	obtains information about a particular token
	C_WaitForSlotEvent	waits for a slot event (token insertion, removal, etc.) to occur
	C_GetMechanismList	obtains a list of mechanisms supported by a token
	C_GetMechanismInfo	obtains information about a particular mechanism
	C_InitToken	initializes a token
	C_InitPIN	initializes the normal user's PIN
	C_SetPIN	modifies the PIN of the current user
Session management	C_OpenSession	opens a connection between an application and a particular token or sets up an

Thanks for listening!