Toward Scaling Hardware Security Module for Emerging Cloud Services

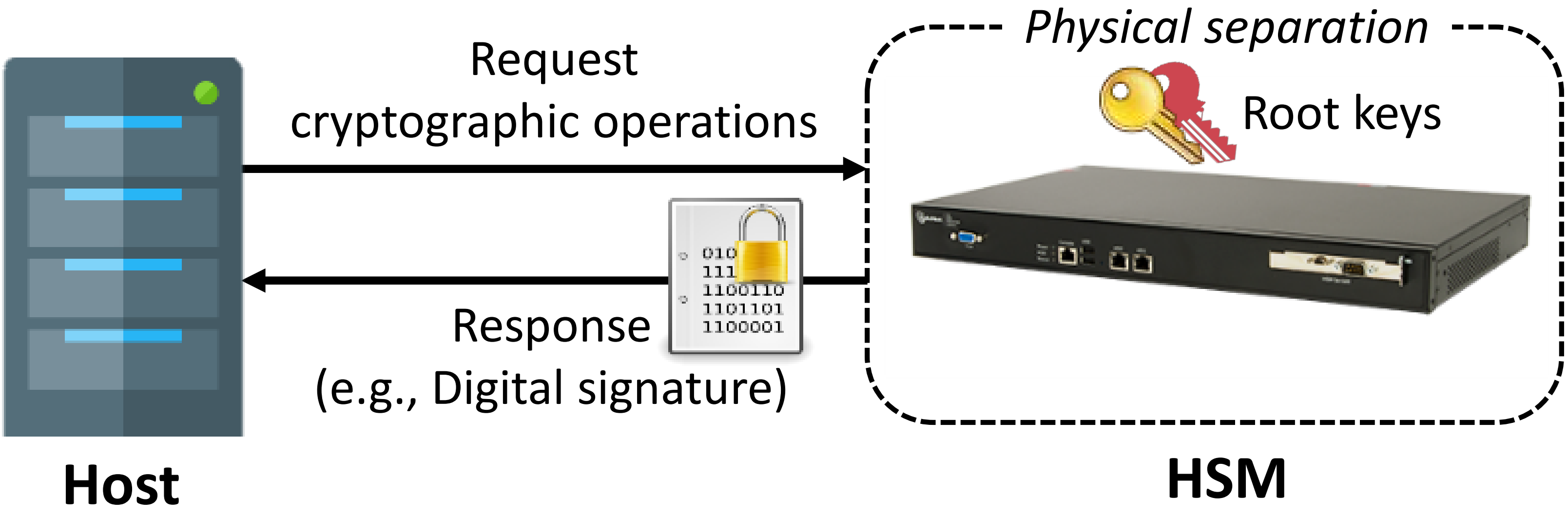
Juhyeng Han*, Seongmin Kim*, Taesoo Kim [†] , Dongsu Han

KAIST [†]Georgia Tech

* The first two authors contributed equally to this work.

Hardware Security Modules (HSMs)

- **Root of trust** for various key management services (KMS)
  - Their root keys should be stored in HSMs
- **Secure physical separation and protection**
- **Satisfies security regulation** requirements such as FIPS 140-2



Physical separation

Root keys

Request
cryptographic operations

Response
(e.g., Digital signature)

Host

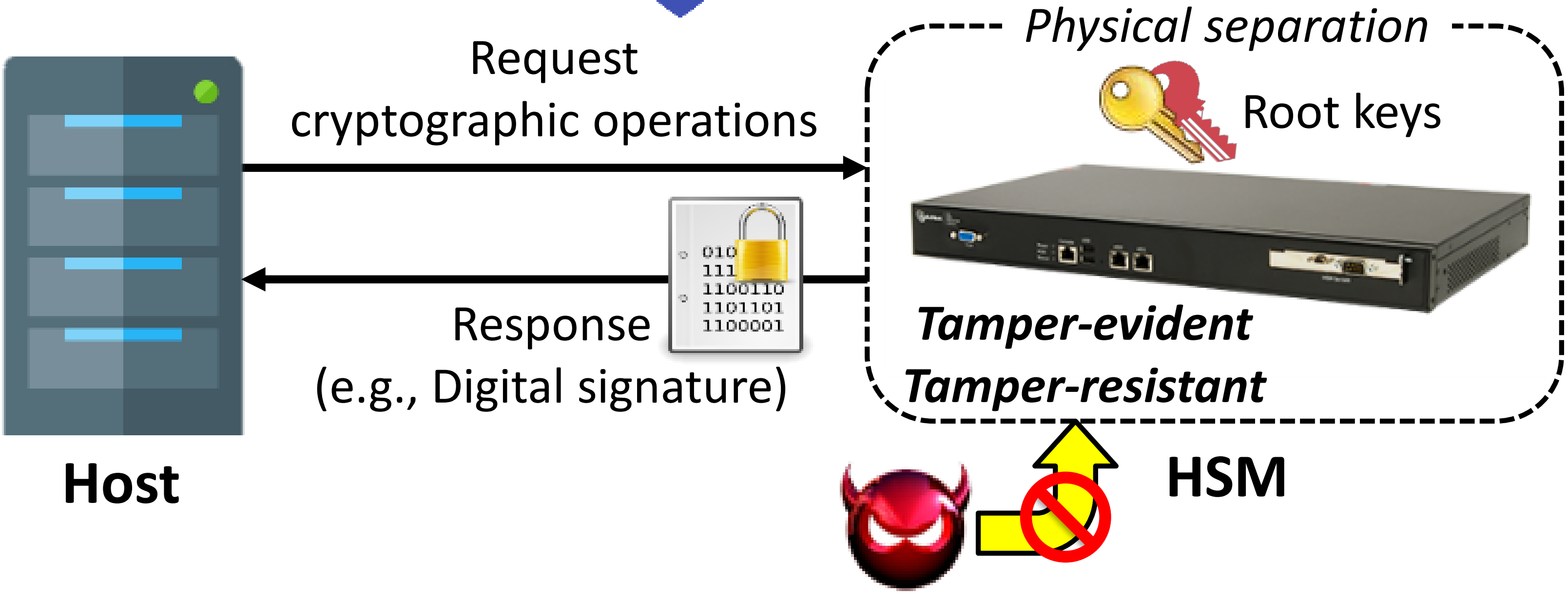
HSM

Hardware Security Modules (HSMs)

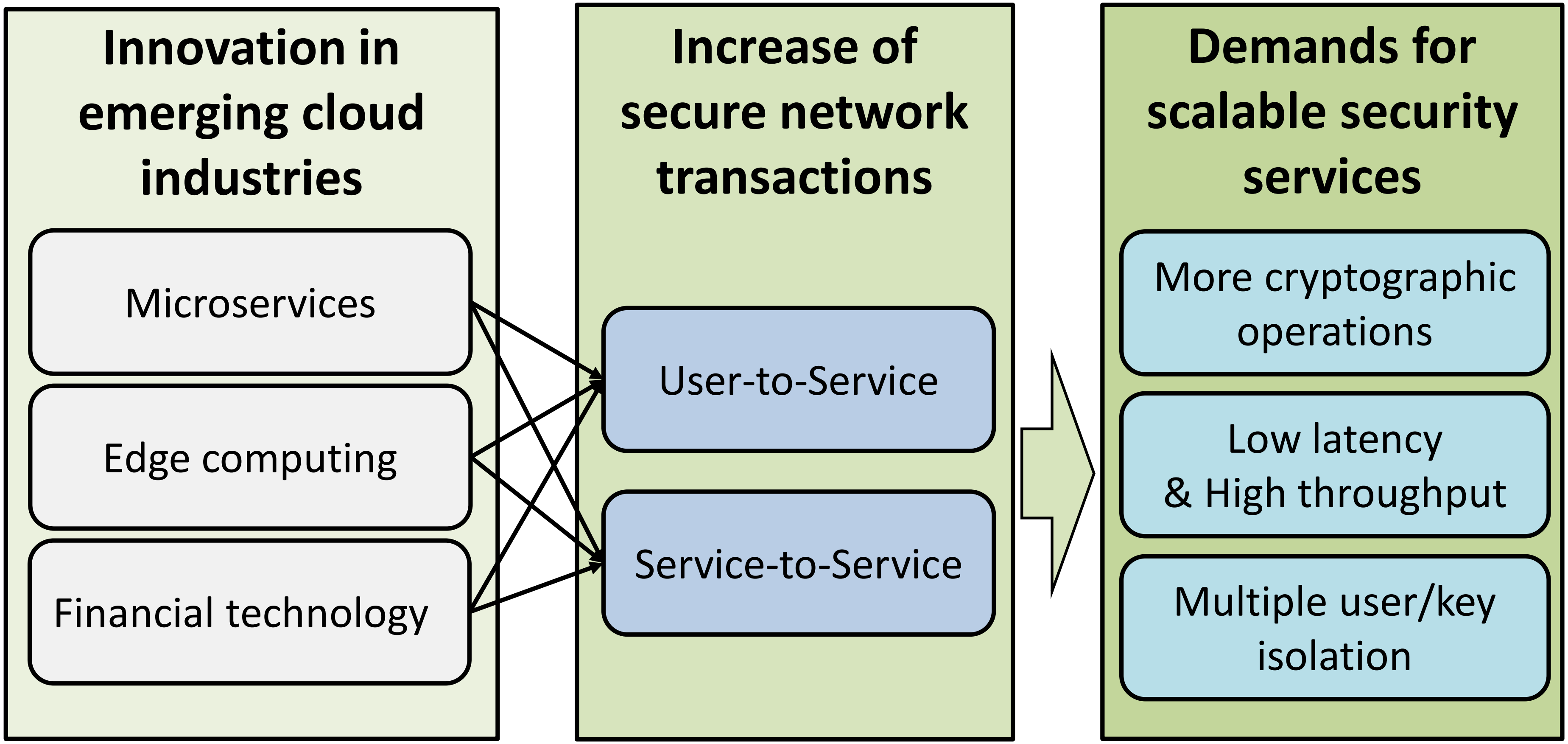
- **Root of trust** for various key management services (KMS)
  - Root keys should be stored in HSMs
- **Secure physical separation and protection**
- **Satisfies security regulation** requirements such as FIPS 140-2

FIPS 140-2

Physical separation required
(U.S. and Canadian security standard)



Demands for Scalable Security Services



Innovation in emerging cloud industries

Microservices

Edge computing

Financial technology

Increase of secure network transactions

User-to-Service

Service-to-Service

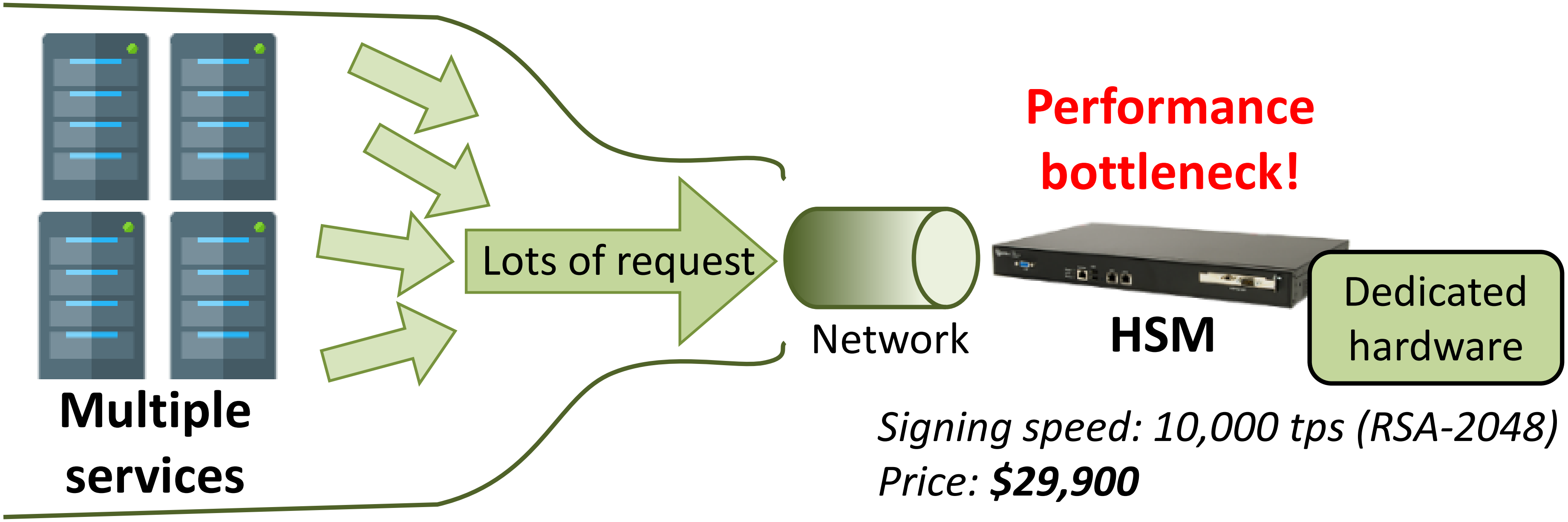
Demands for scalable security services

More cryptographic operations

Low latency
& High throughput

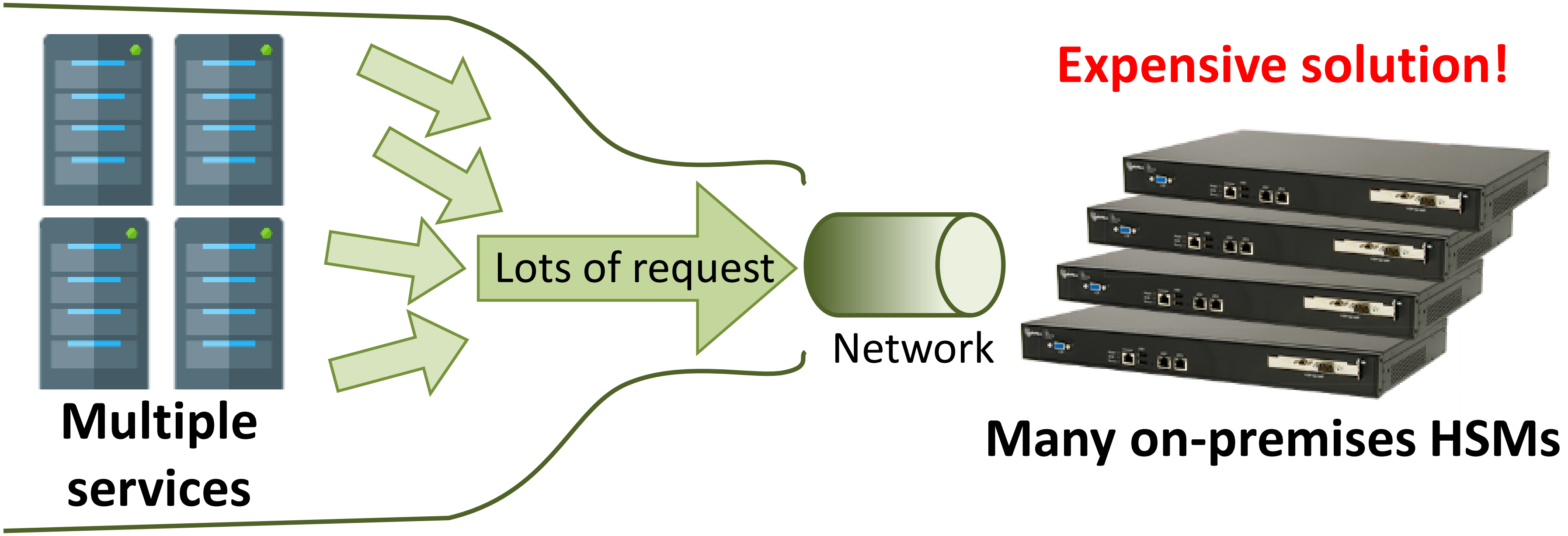
Multiple user/key isolation

Problem: Limited Scalability of HSMs



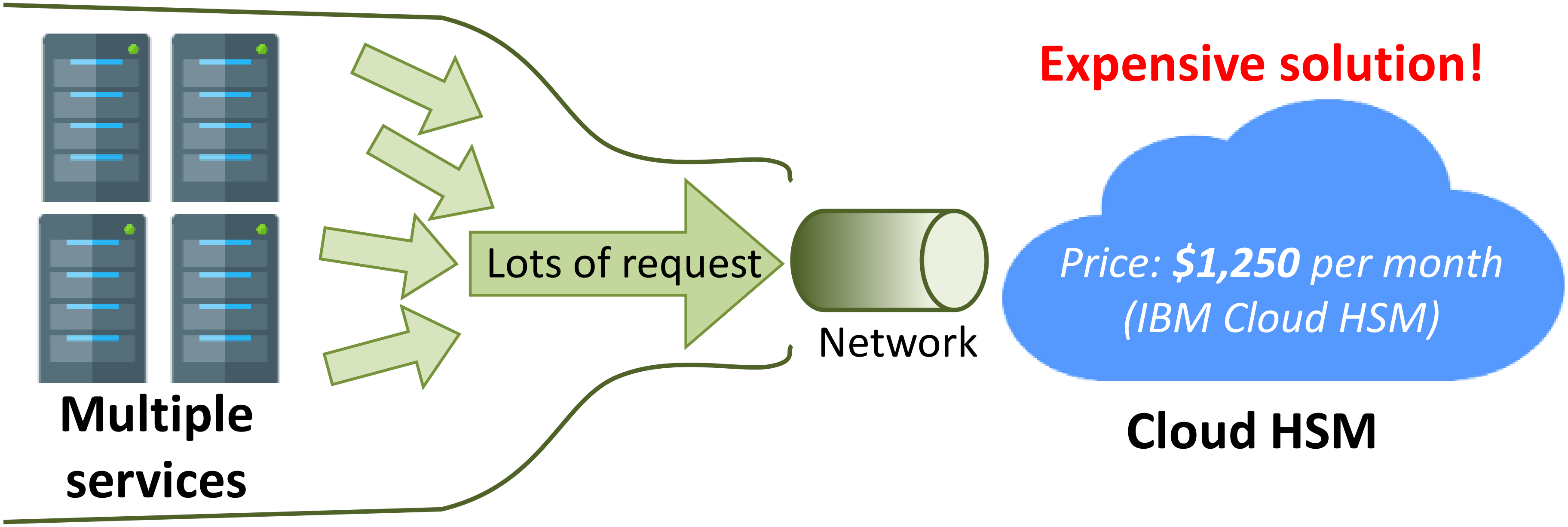
Signing speed: 10,000 tps (RSA-2048)
Price: ***$29,900***

Problem: Limited Scalability of HSMs



Expensive solution!

Problem: Limited Scalability of HSMs



Expensive solution!

Multiple services

Cloud HSM

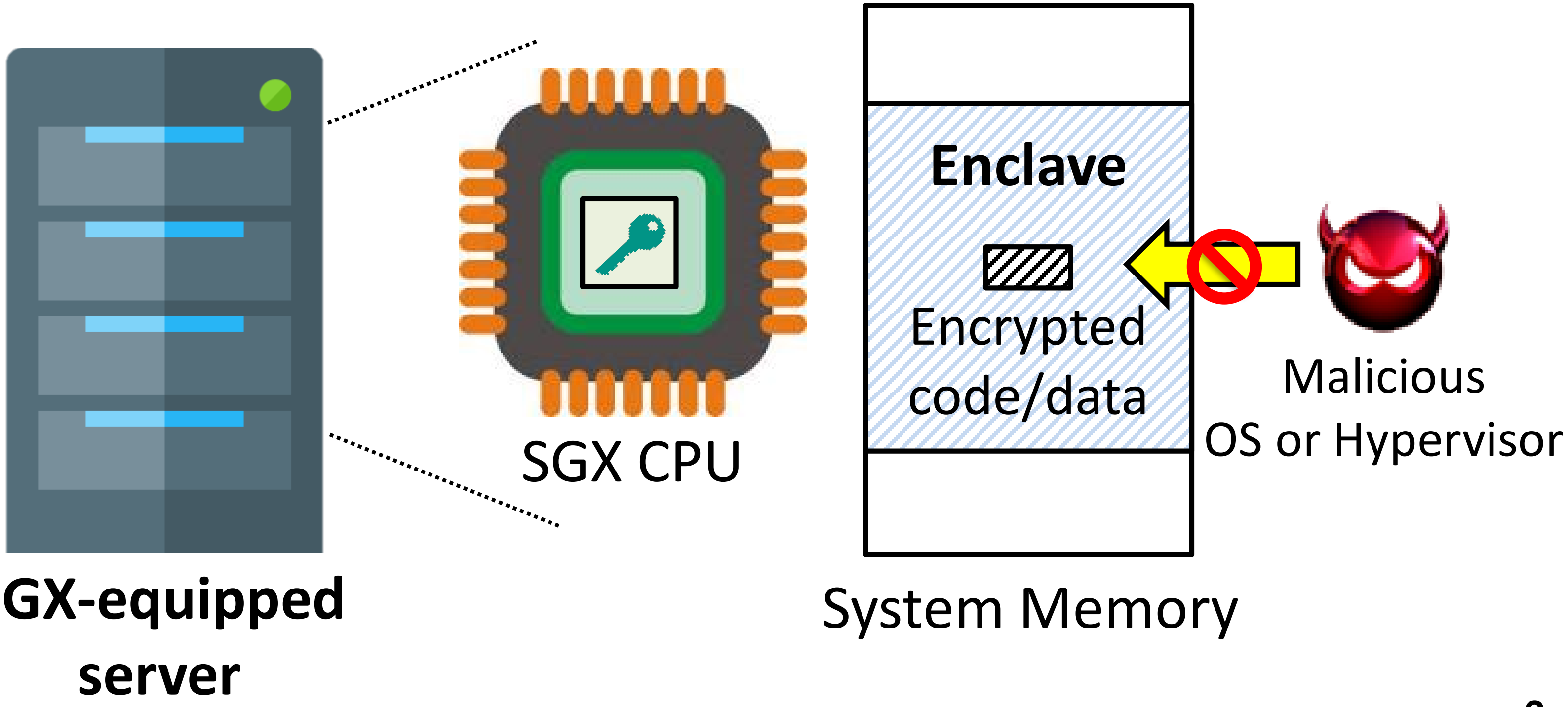
Problem: Limited Scalability of HSMs

Can we efficiently scale out HSMs for key management services?

Alternative Approach

- Leverages commodity Trusted Execution Environment (TEE) instead of HSMs

  [S. Chakrabarti et al. "Intel® SGX Enabled Key Manager Service with OpenStack Barbican." arXiv preprint arXiv:1712.07694, 2017.]

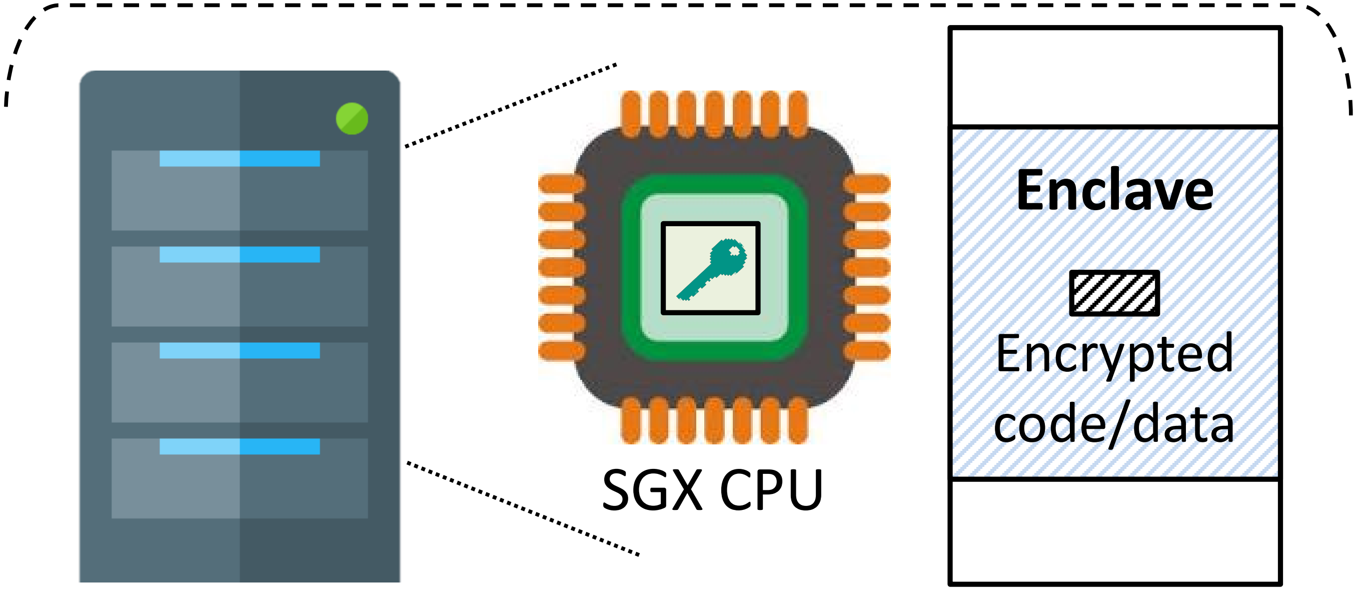


Limitation of the Alternative Approach

- Leverages commodity Trusted Execution Environment (TEE) instead of HSMs

  [S. Chakrabarti et al. “Intel® SGX Enabled Key Manager Service with OpenStack Barbican.” arXiv preprint arXiv:1712.07694, 2017.]

Does not provide physical separation & protection



SGX CPU

Enclave

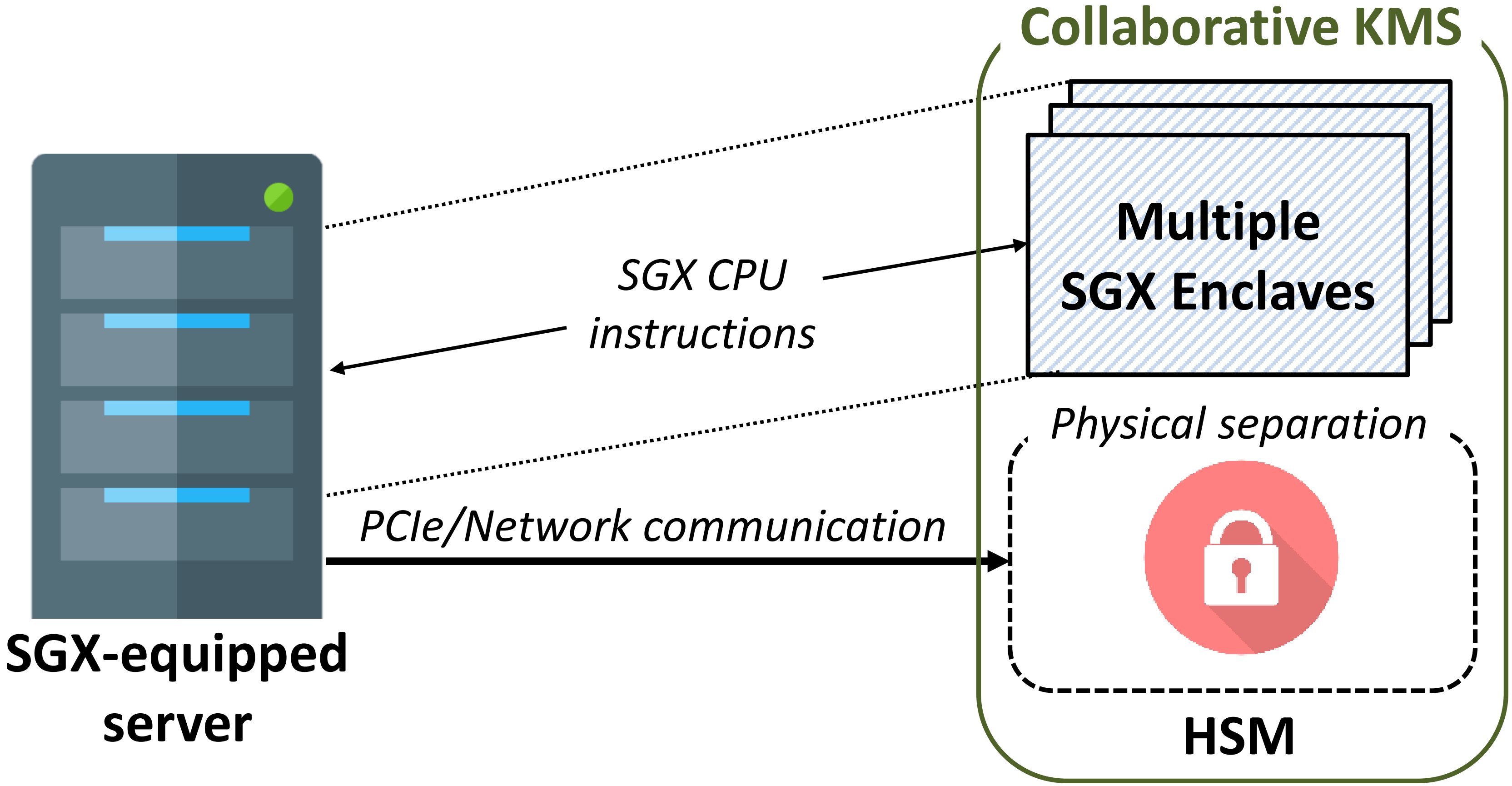
Encrypted code/data

SGX-equipped server

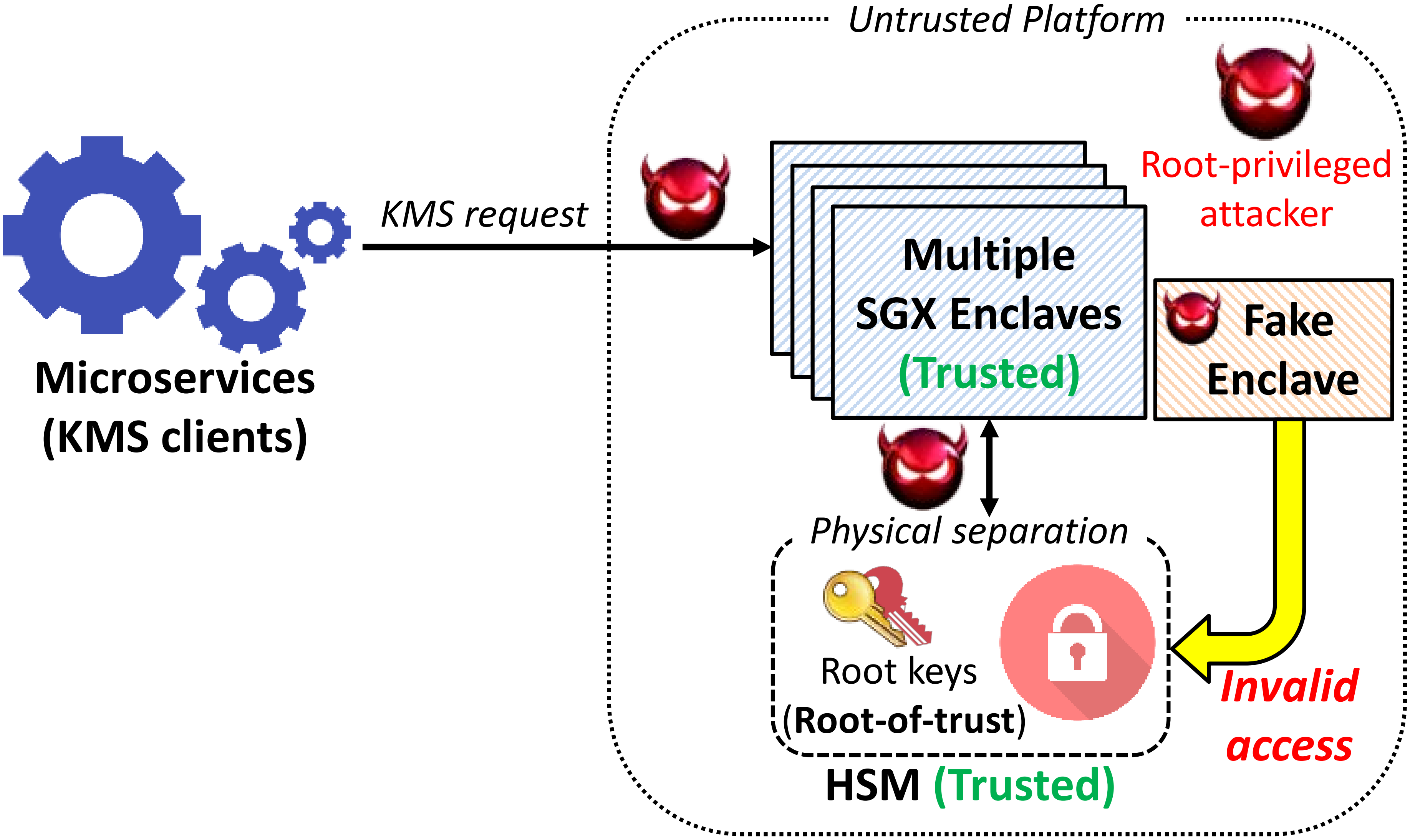
System Memory

Approach : Combining HSMs with TEE-based KMS

- Achieves cost-efficient scalability with SGX technology
- Maintains security level of physical separation with HSMs
- SGX enclaves and HSMs collaborate for key management

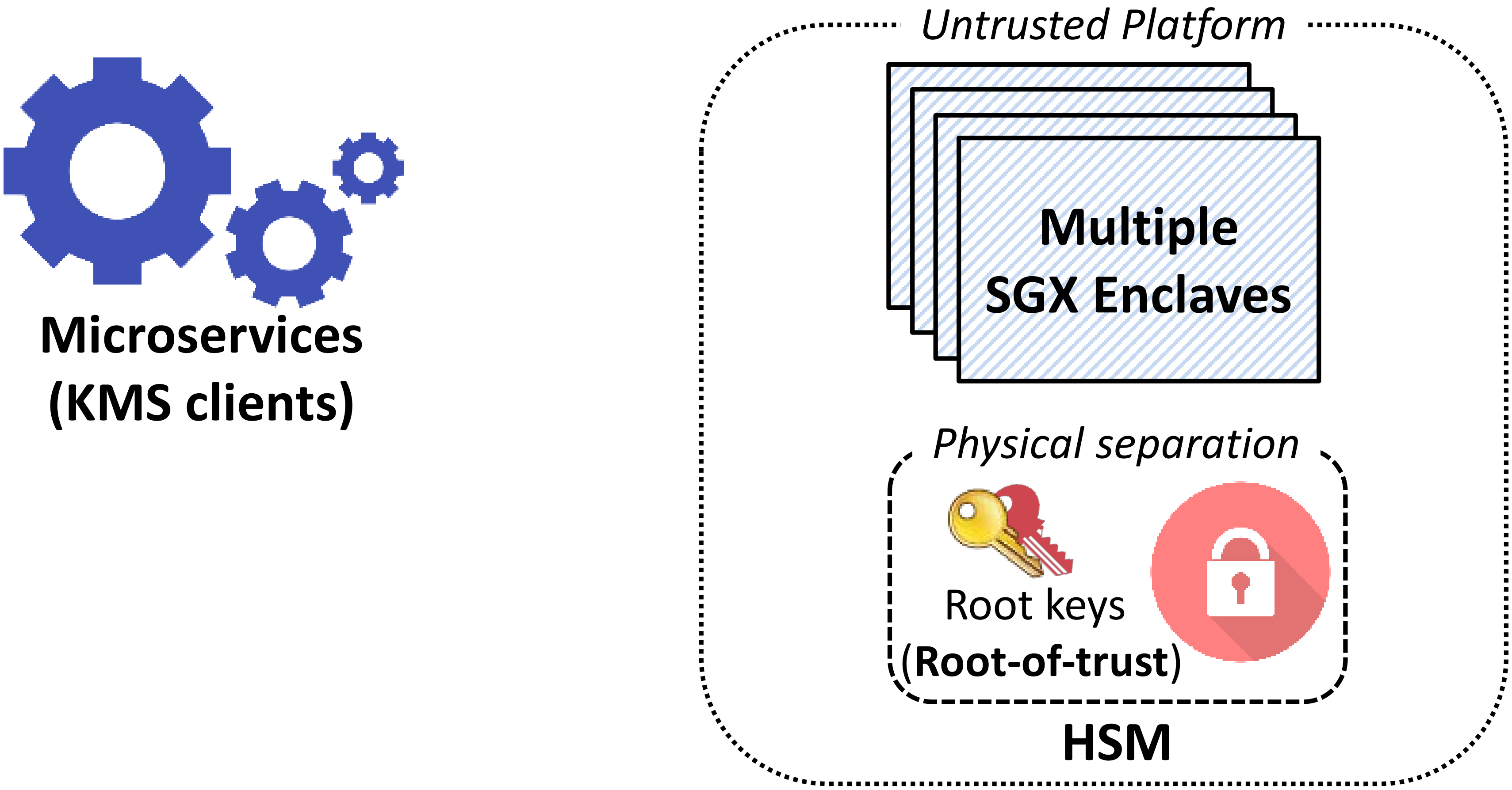


Deployment Assumption & Threat Model



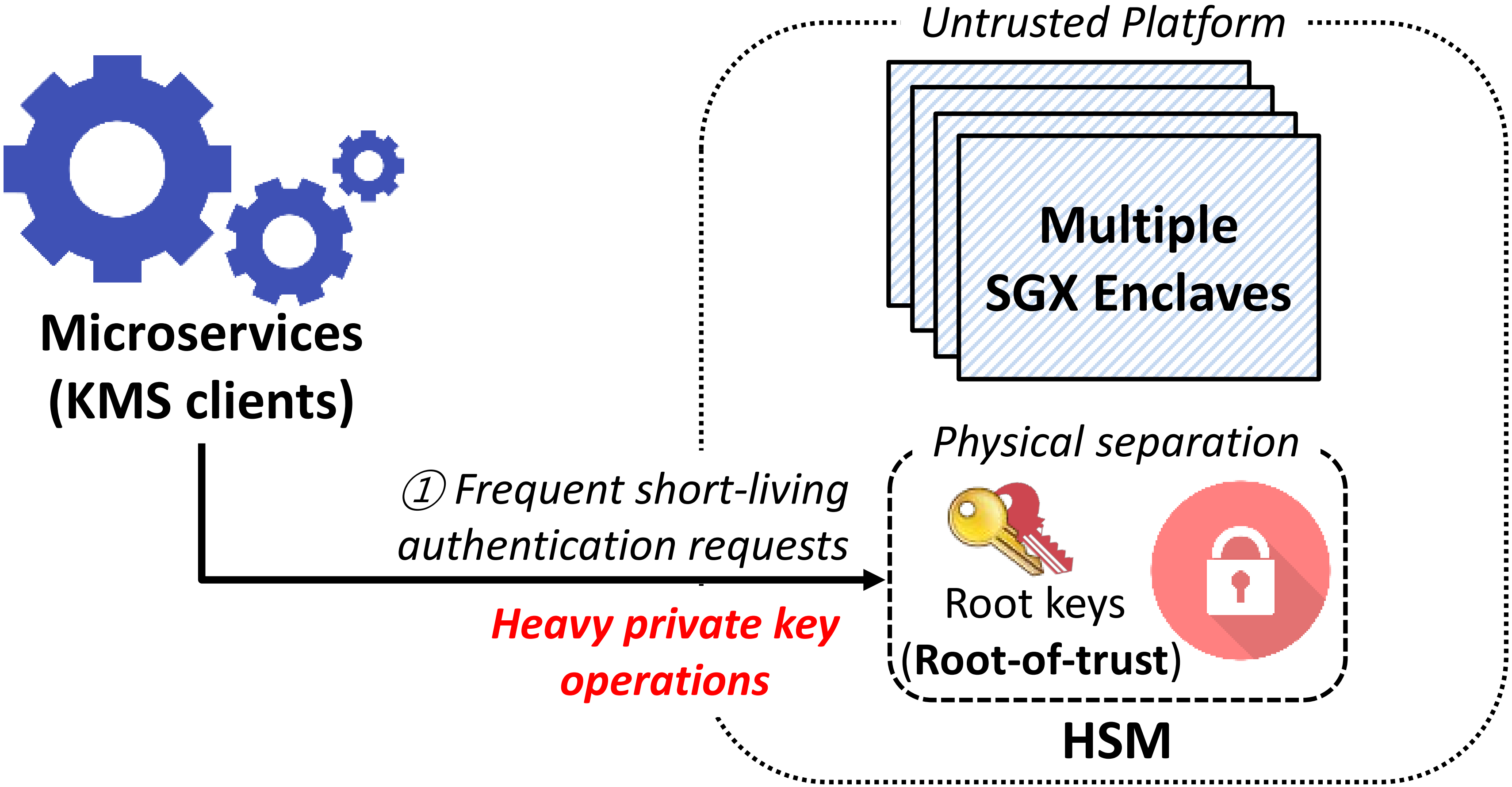
Challenge 1 : Scaling Performance

- Frequent private key operation requests to HSMs can incur performance bottleneck.



Challenge 1 : Scaling Performance

- Frequent private key operation requests to HSMs can incur performance bottleneck.



Untrusted Platform

Multiple SGX Enclaves

Microservices (KMS clients)

① Frequent short-living authentication requests

Heavy private key operations

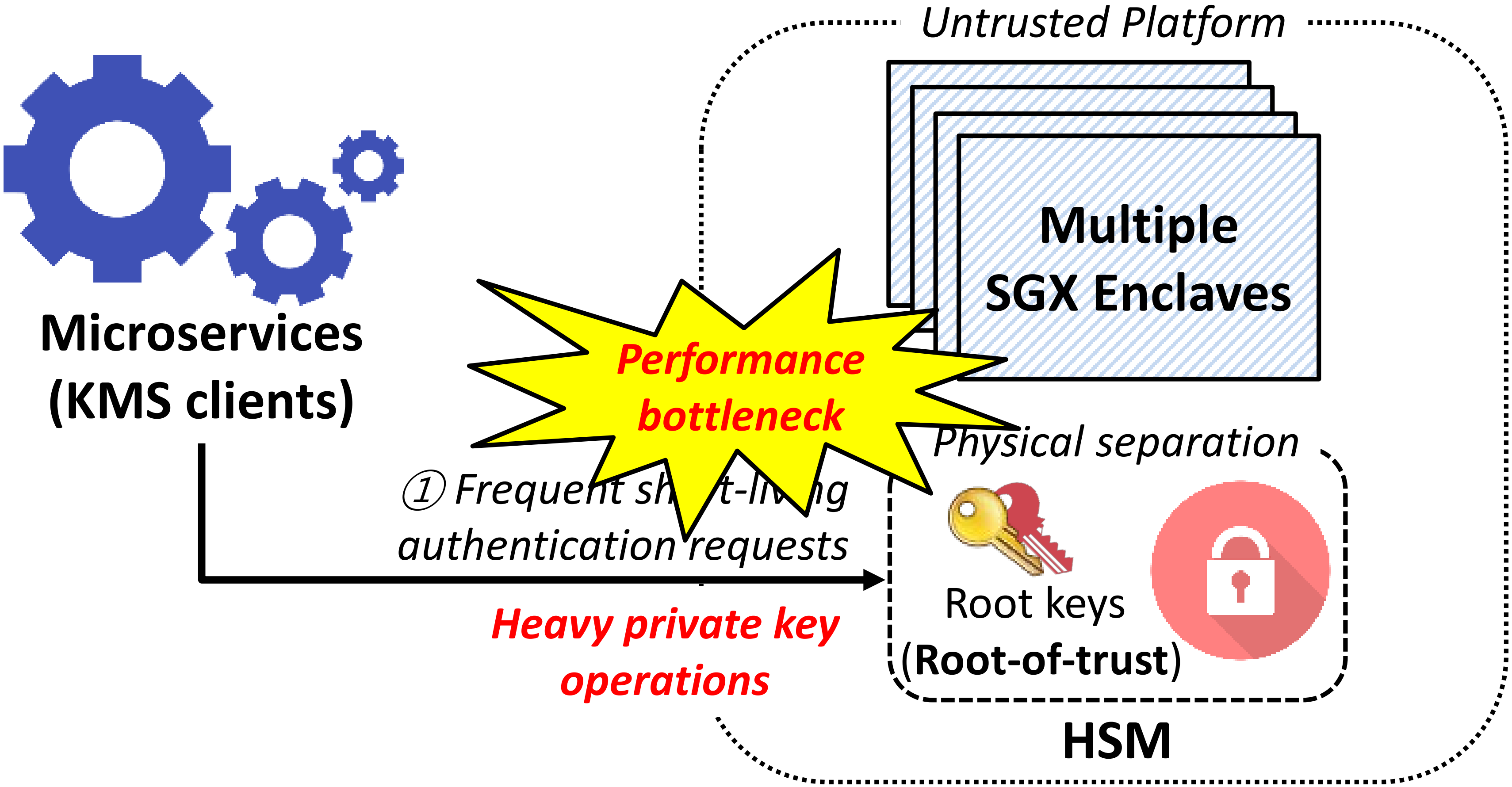
Physical separation

Root keys **(Root-of-trust)**

HSM

Challenge 1 : Scaling Performance

- Frequent private key operation requests to HSMs can incur performance bottleneck.



Untrusted Platform

Multiple SGX Enclaves

Microservices (KMS clients)

Performance bottleneck

① *Frequent short-living authentication requests*

Heavy private key operations

Physical separation

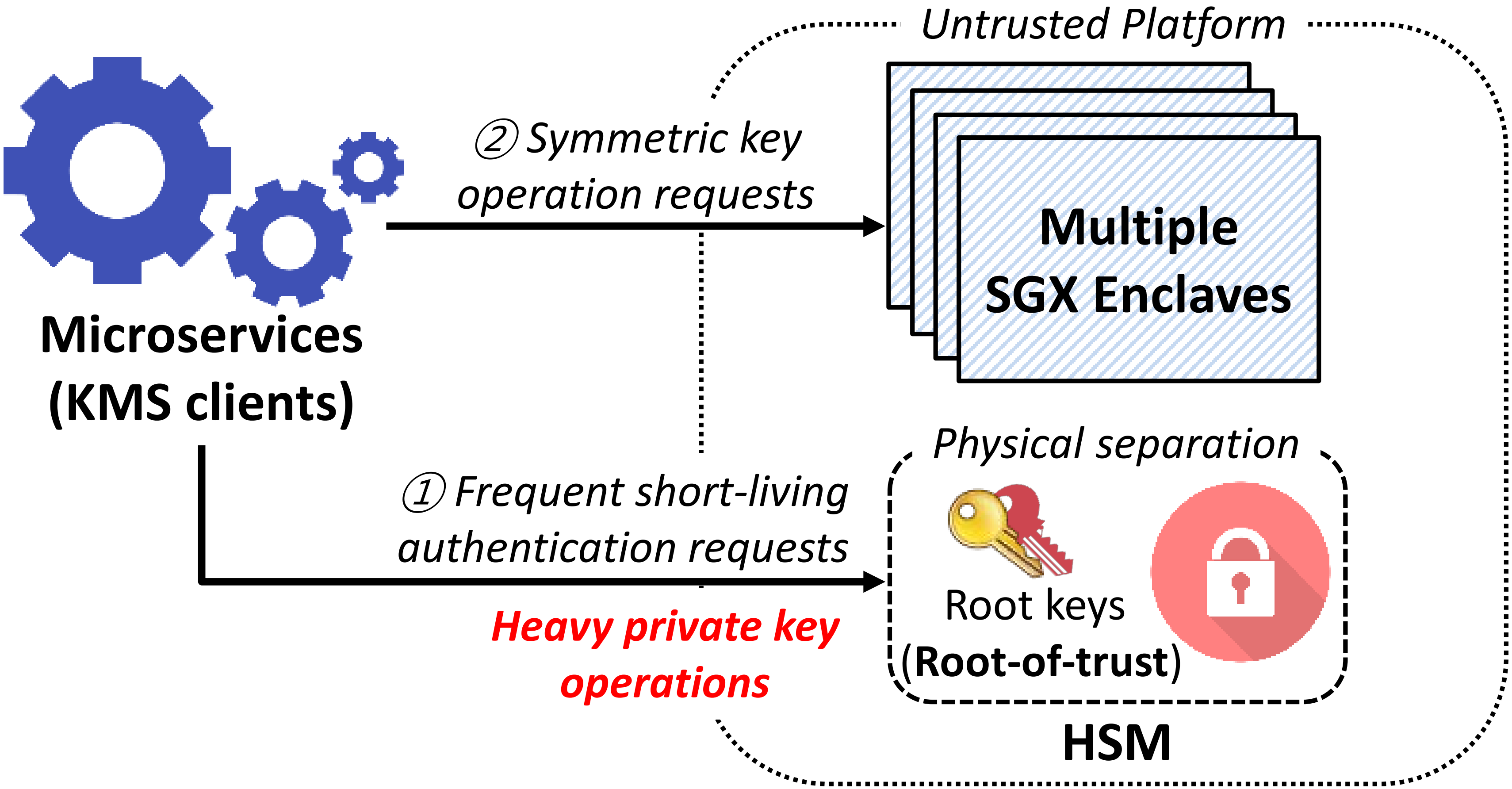
Root keys

(Root-of-trust)

HSM

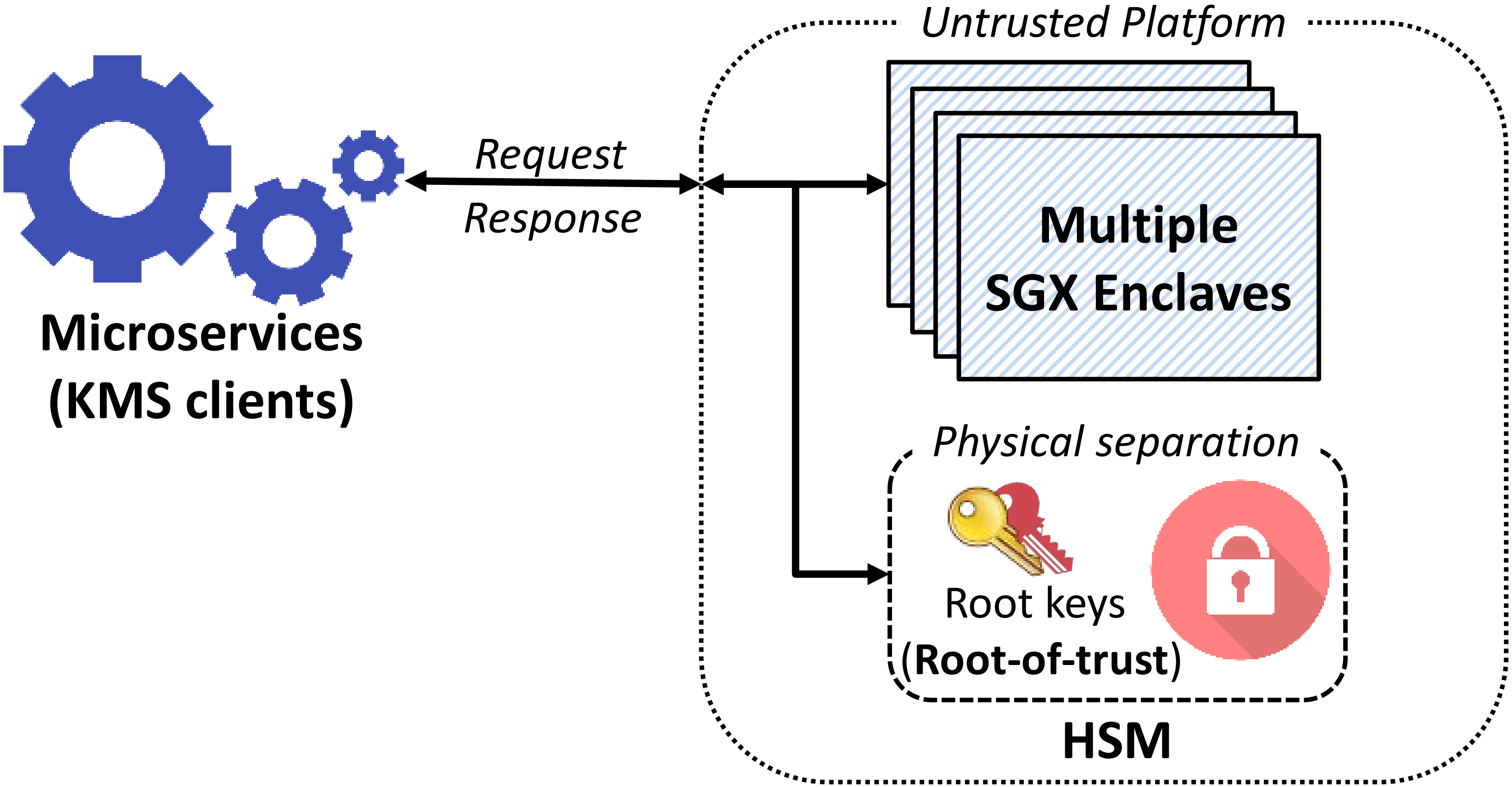
Challenge 1 : Scaling Performance

- Frequent private key operation requests to HSMs can incur performance bottleneck.



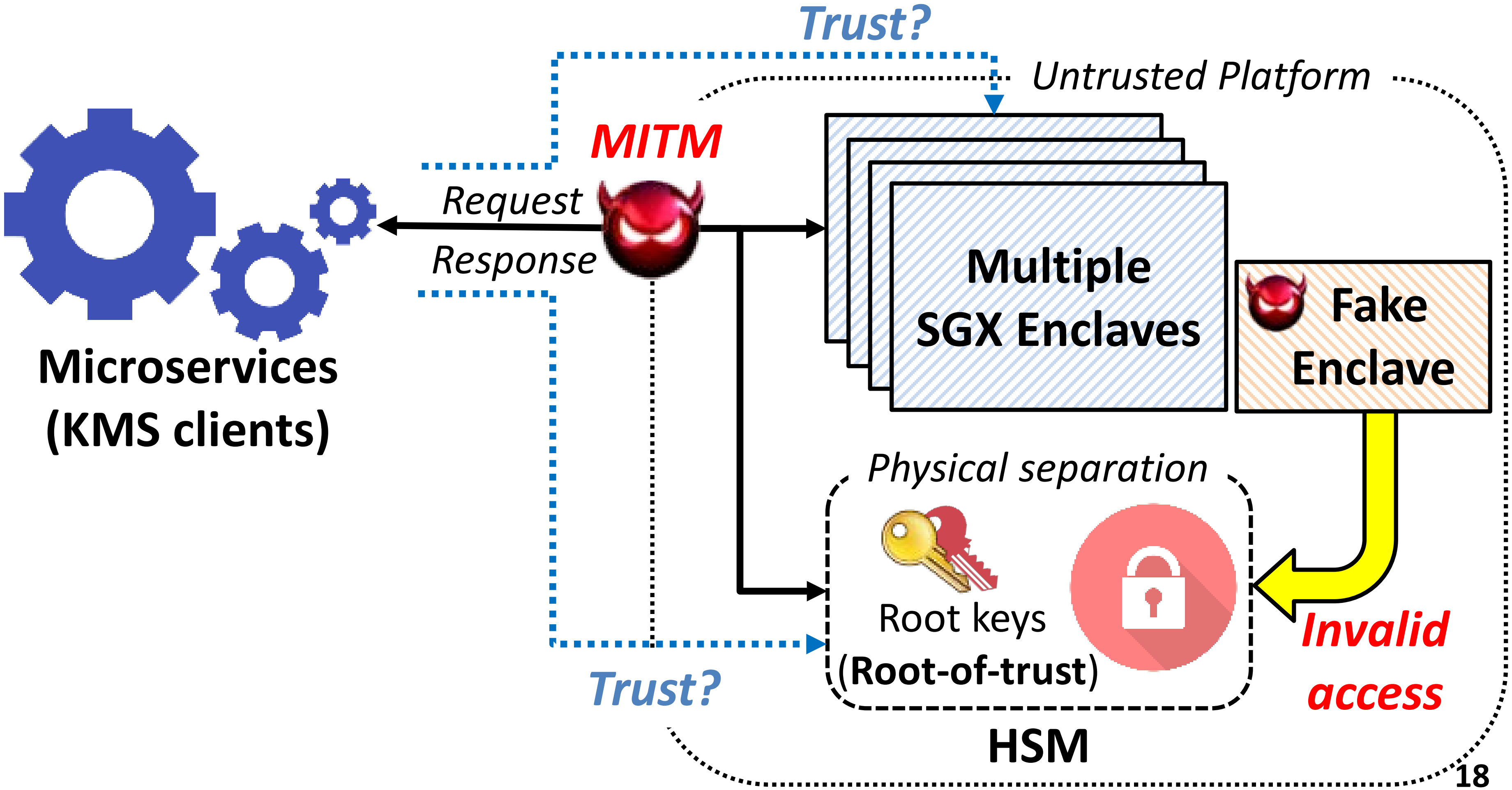
Challenge 2 : Validation between Enclaves and HSMs

- KMS clients, SGX enclaves and HSMs should trust each others
- Lack of validation mechanism between SGX enclaves and HSMs



Challenge 2 : Validation between Enclaves and HSMs

- KMS clients, SGX enclaves and HSMs should trust each others
- Lack of validation mechanism between SGX enclaves and HSMs



Design Goals of ScaleTrust

1. **Scalable performance**

   Enhances performance by scaling out and does not make an HSM a performance bottleneck

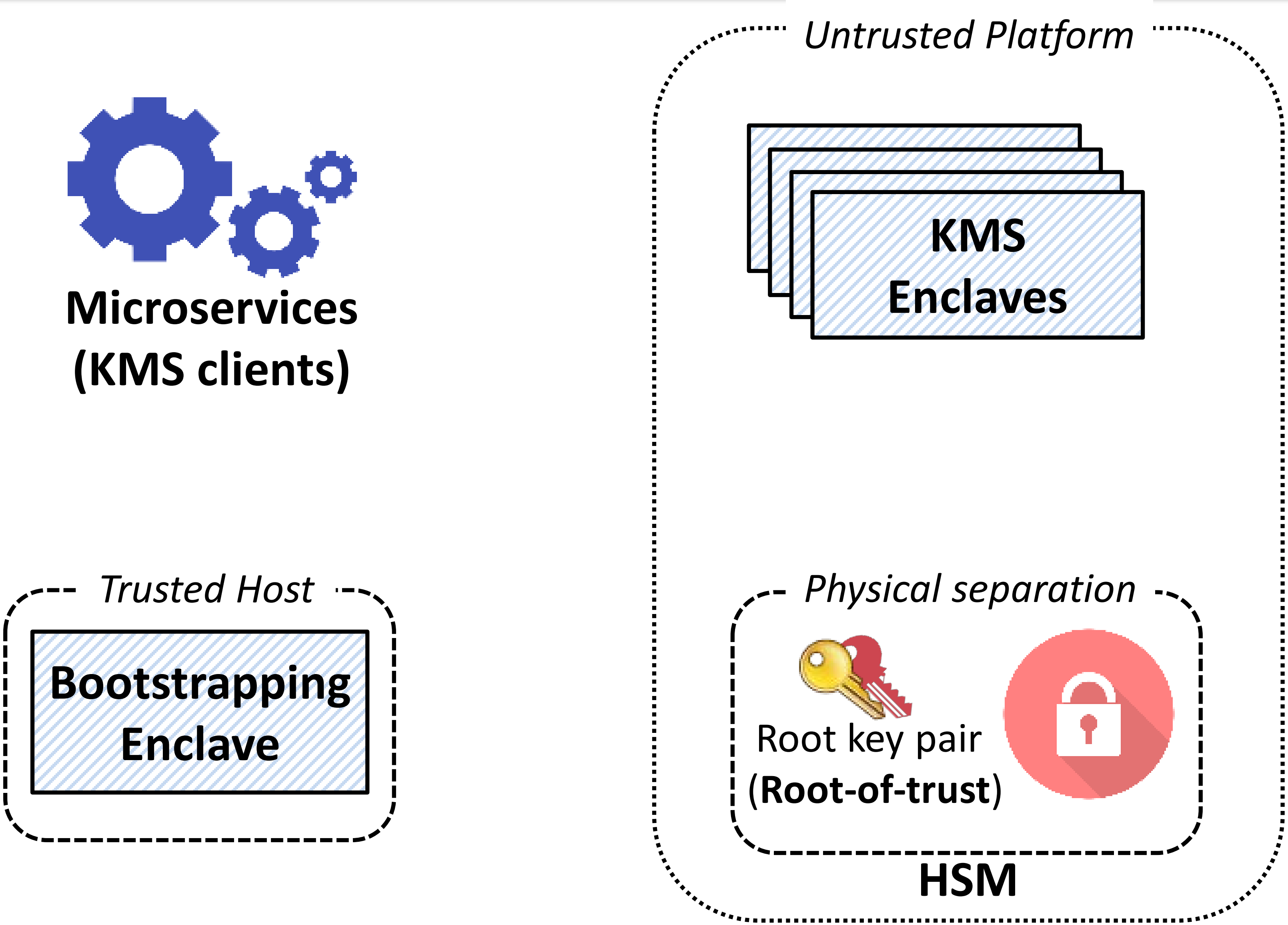
2. **Cost-effectiveness**

   Cost-efficiently scales out for key management services

3. **Security**

   Preserves a chain-of-trust from an HSM to clients

Design Overview



Microservices
(KMS clients)

Trusted Host

Bootstrapping Enclave

Untrusted Platform

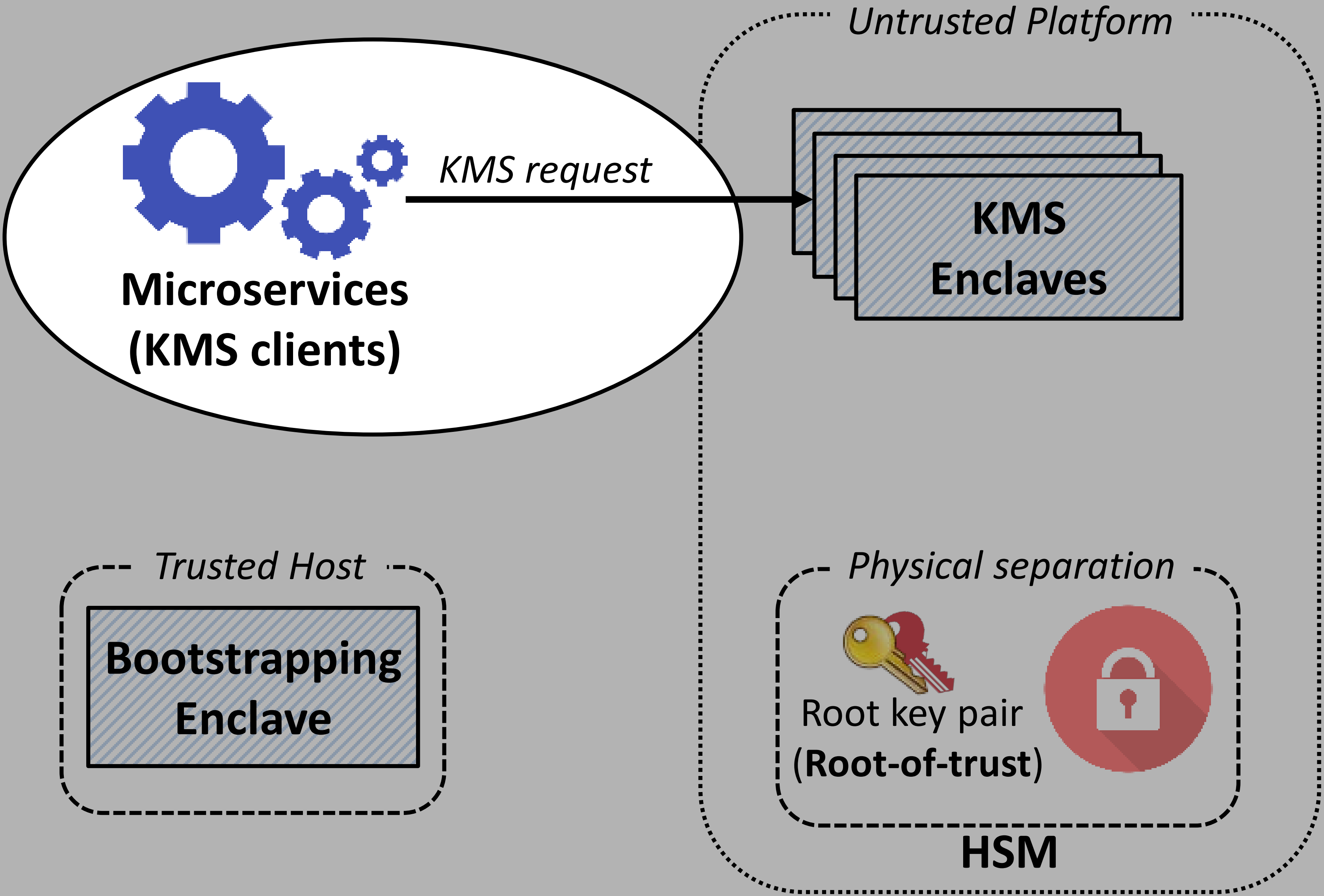
KMS Enclaves

Physical separation

Root key pair
(Root-of-trust)

HSM

Design Overview



Untrusted Platform

Microservices
(KMS clients)

KMS request

KMS
Enclaves

Trusted Host

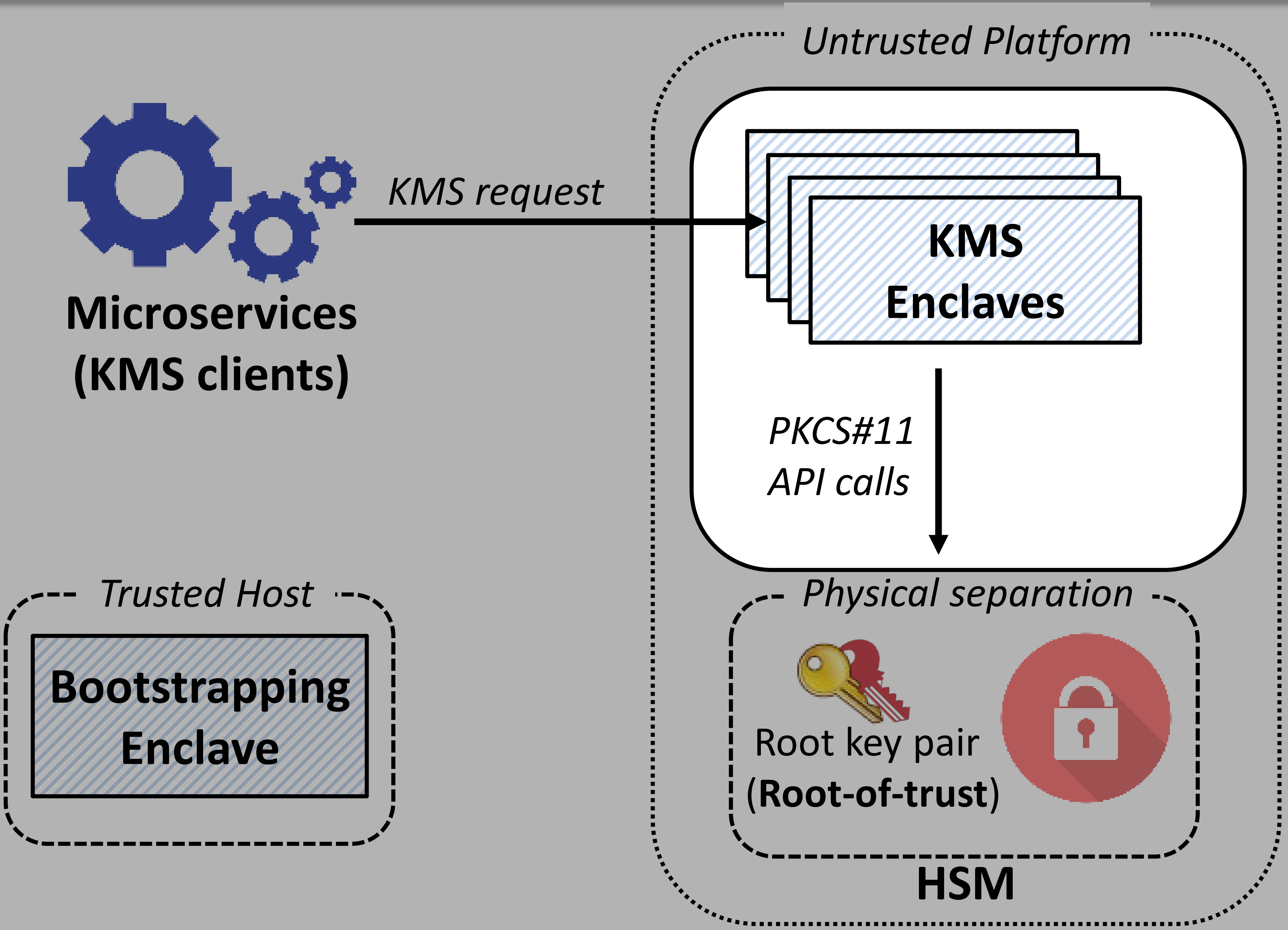
Bootstrapping
Enclave

Physical separation

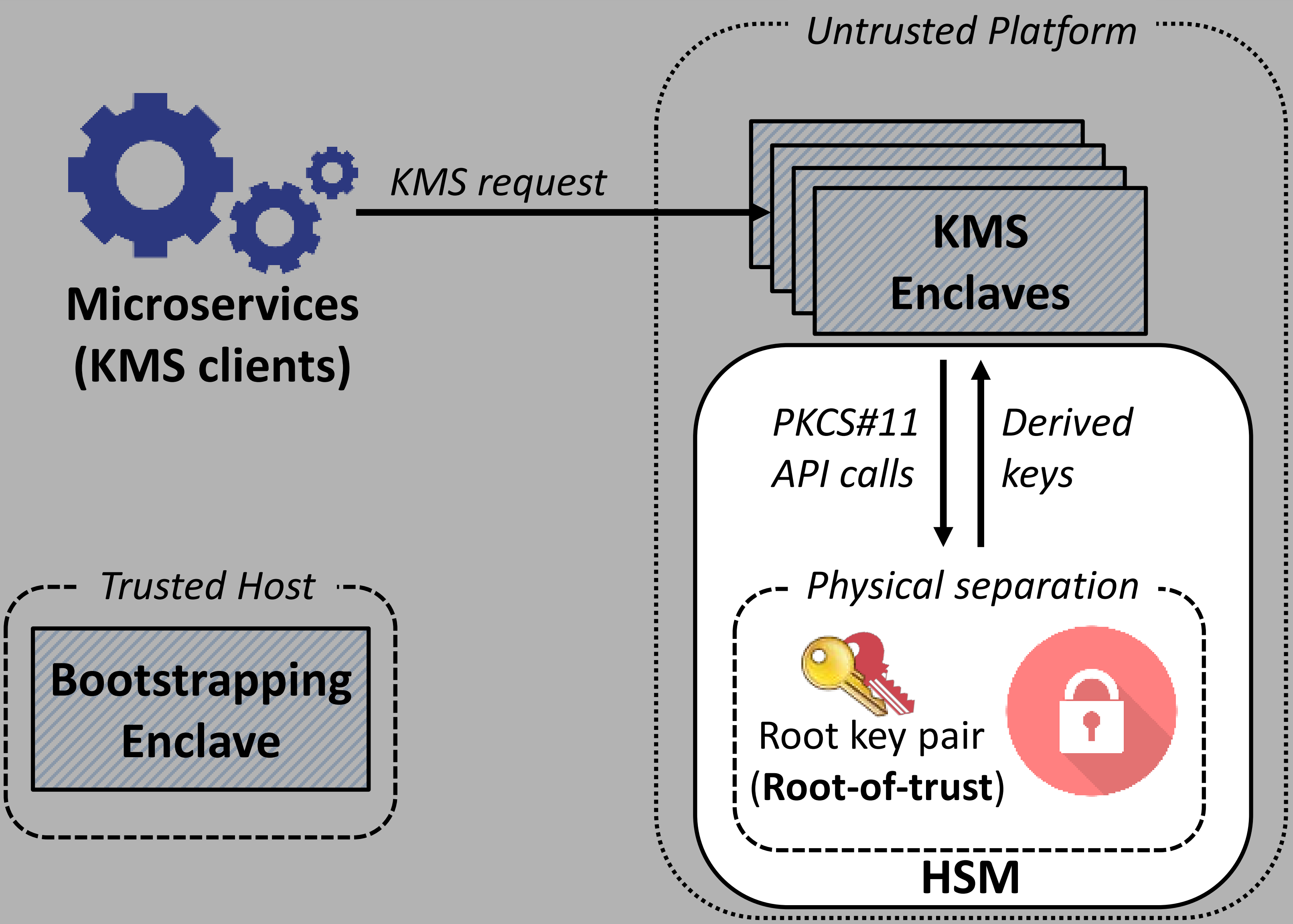
Root key pair
(Root-of-trust)

HSM

Design Overview



Design Overview



Untrusted Platform

Microservices (KMS clients)

KMS request

KMS Enclaves

PKCS#11 API calls

Derived keys

Physical separation

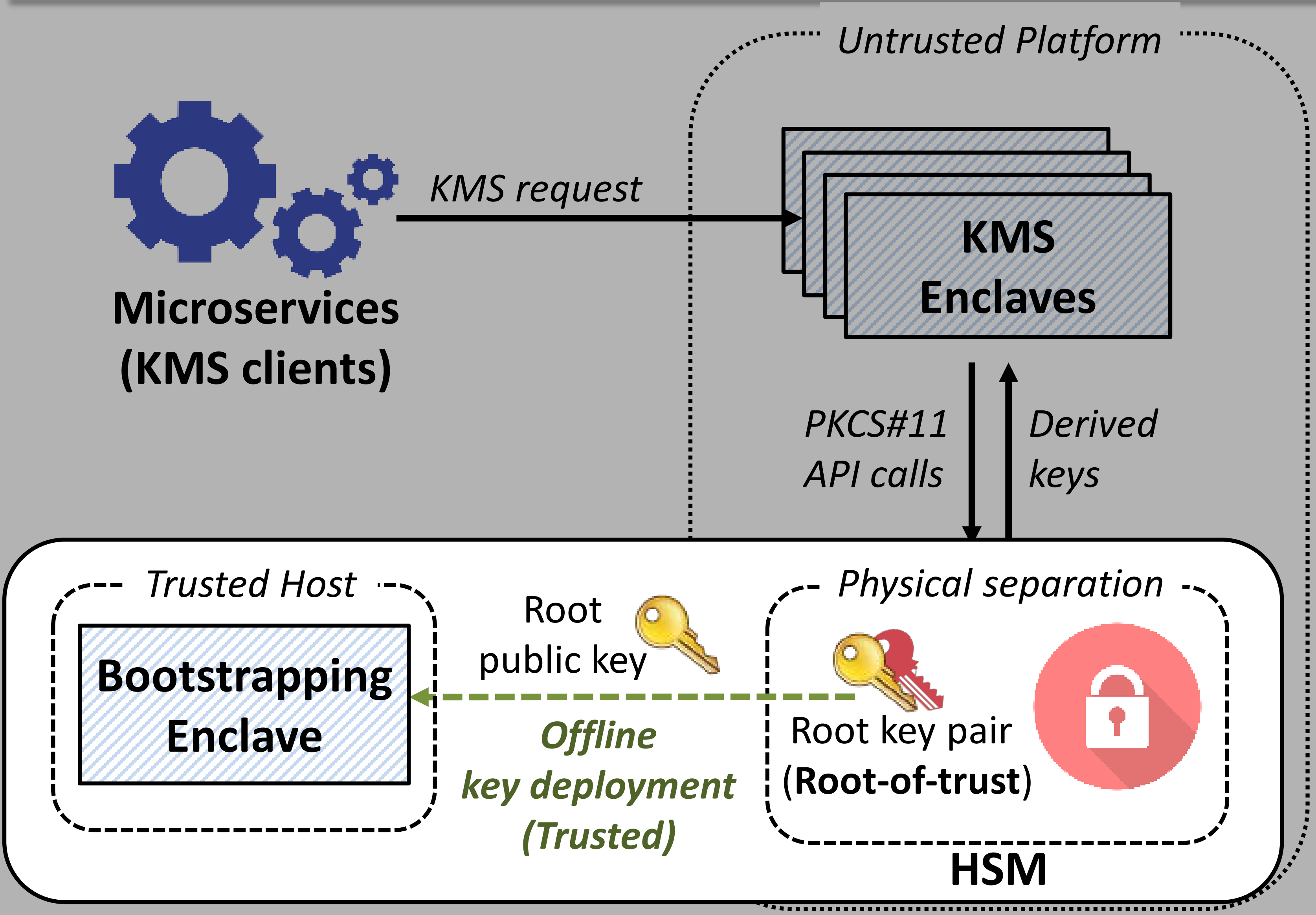
Root key pair **(Root-of-trust)**

HSM

Trusted Host

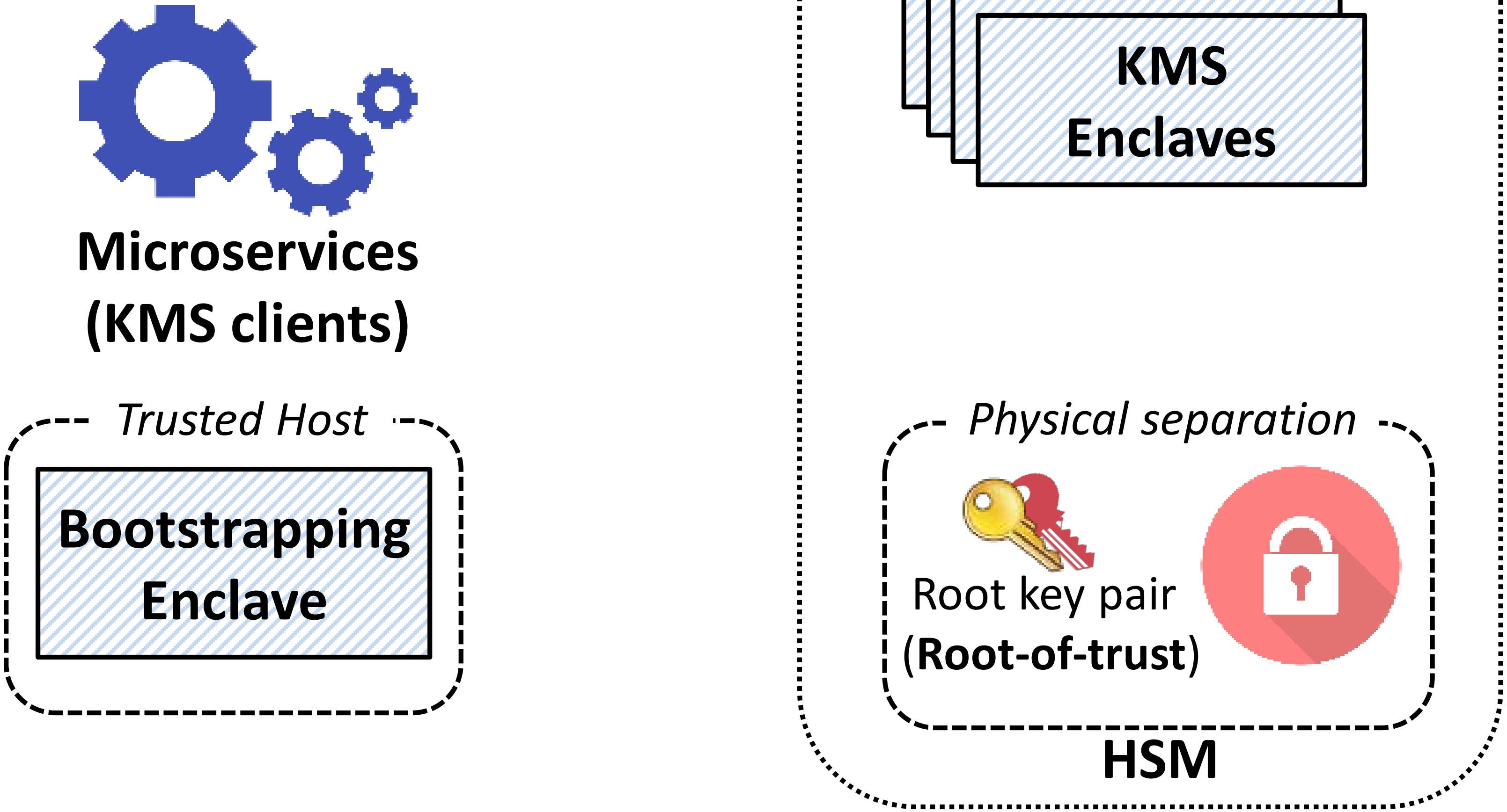
Bootstrapping Enclave

Design Overview



Secure bootstrapping

Secure bootstrapping ① :
An HSM generates
a root key pairs



Microservices
(KMS clients)

Trusted Host

Bootstrapping Enclave

Untrusted Platform

KMS Enclaves

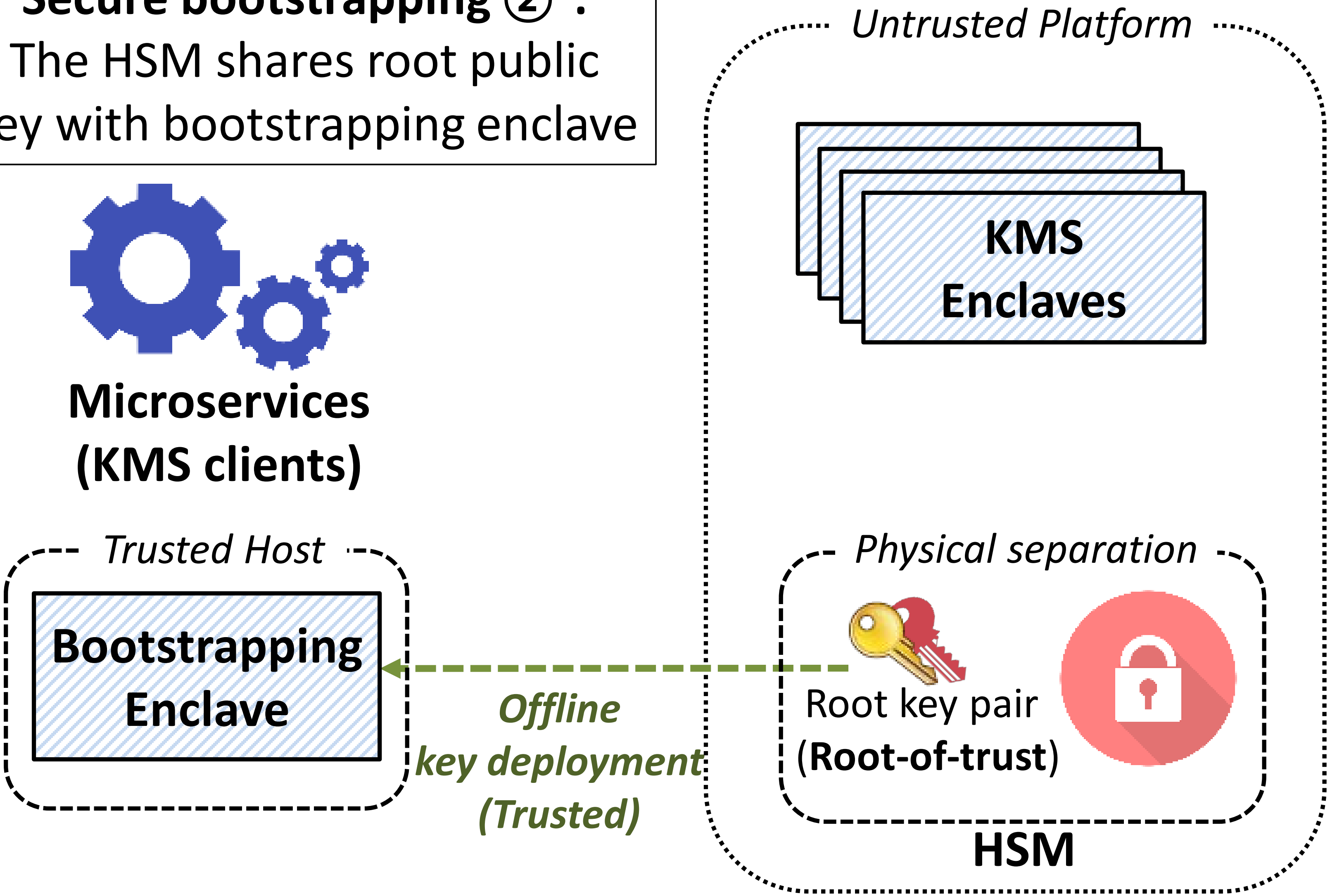
Physical separation

Root key pair
(Root-of-trust)

HSM

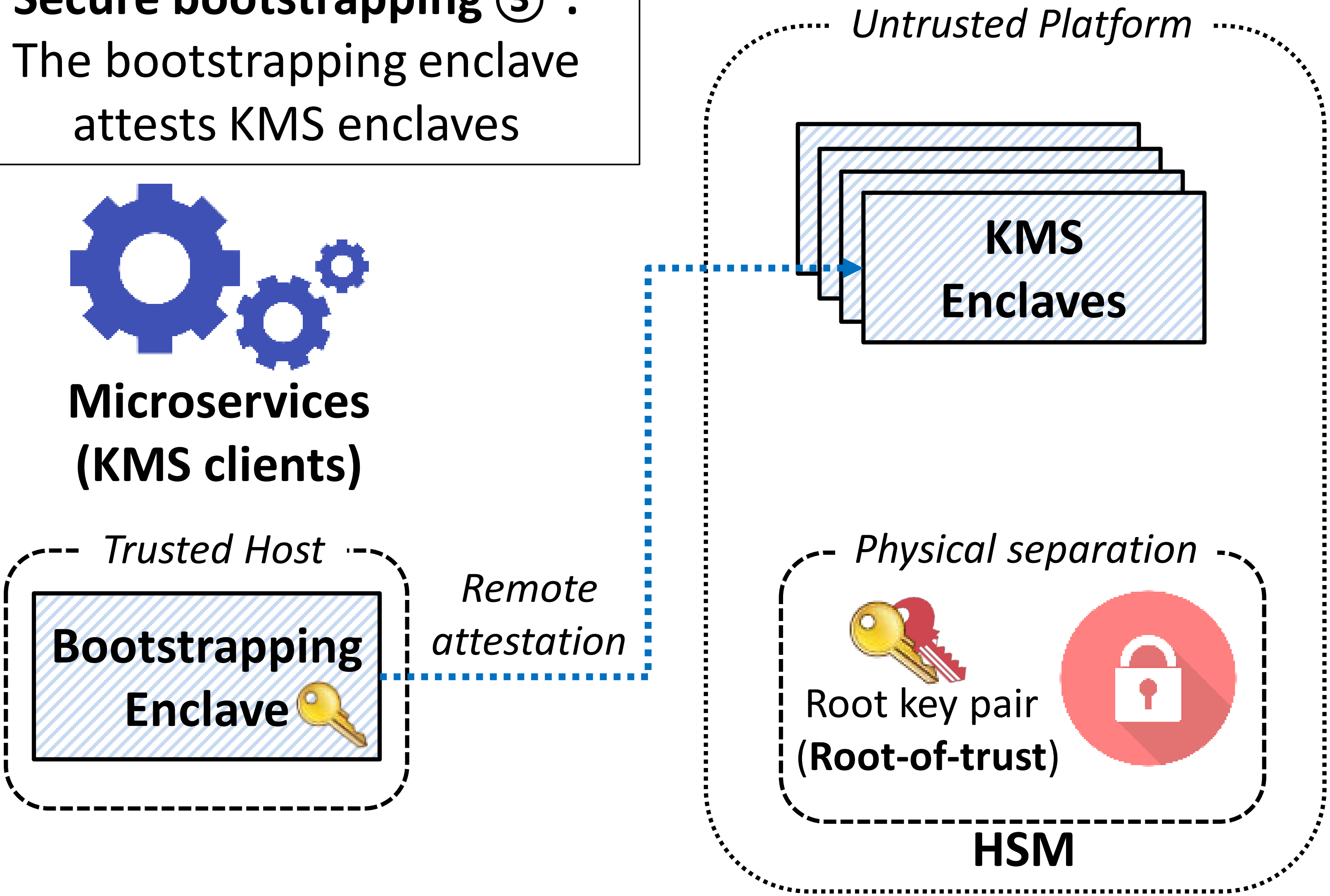
Secure bootstrapping

Secure bootstrapping ② :
The HSM shares root public key with bootstrapping enclave



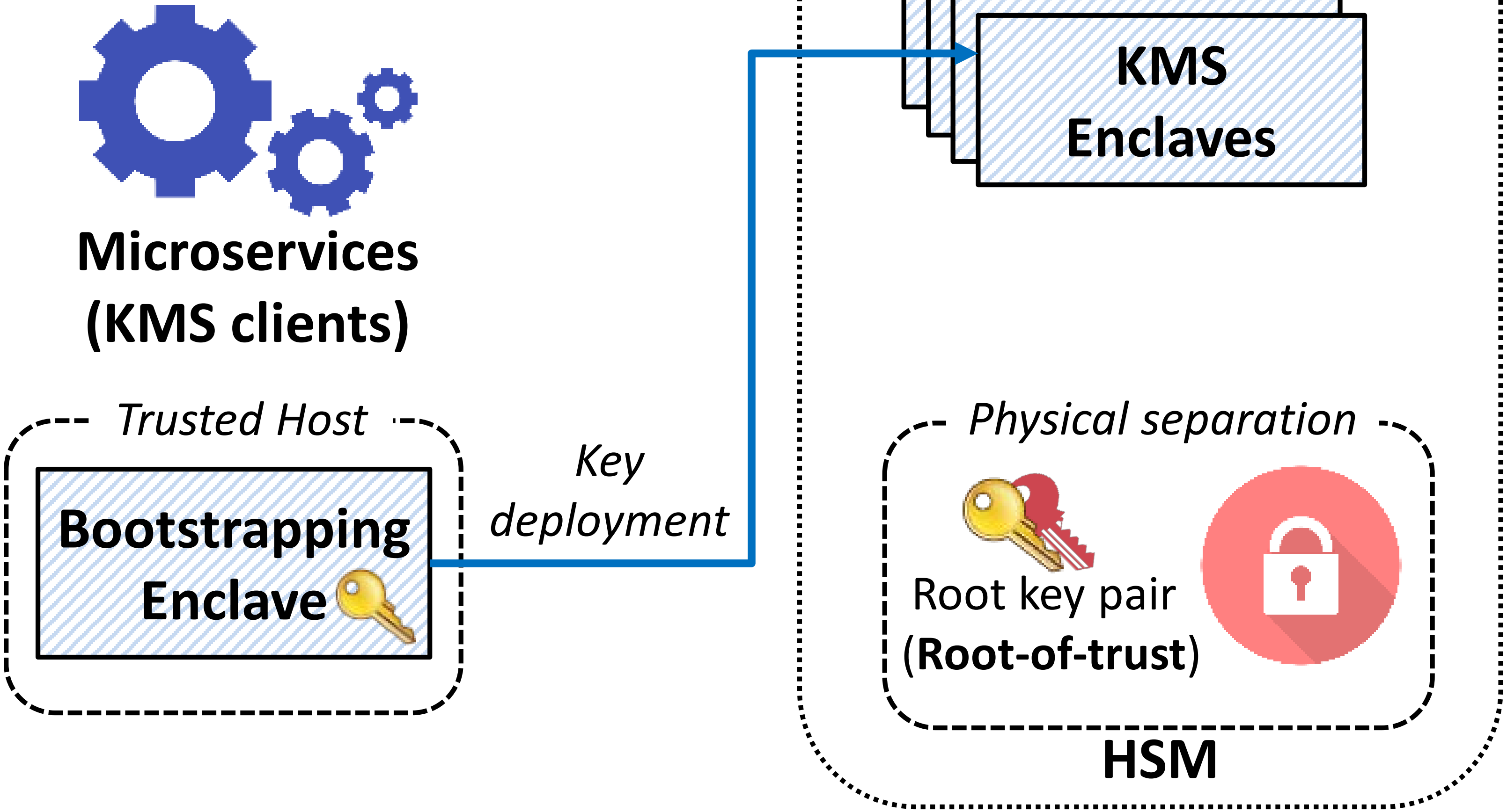
Secure bootstrapping

Secure bootstrapping ③ :
The bootstrapping enclave
attests KMS enclaves



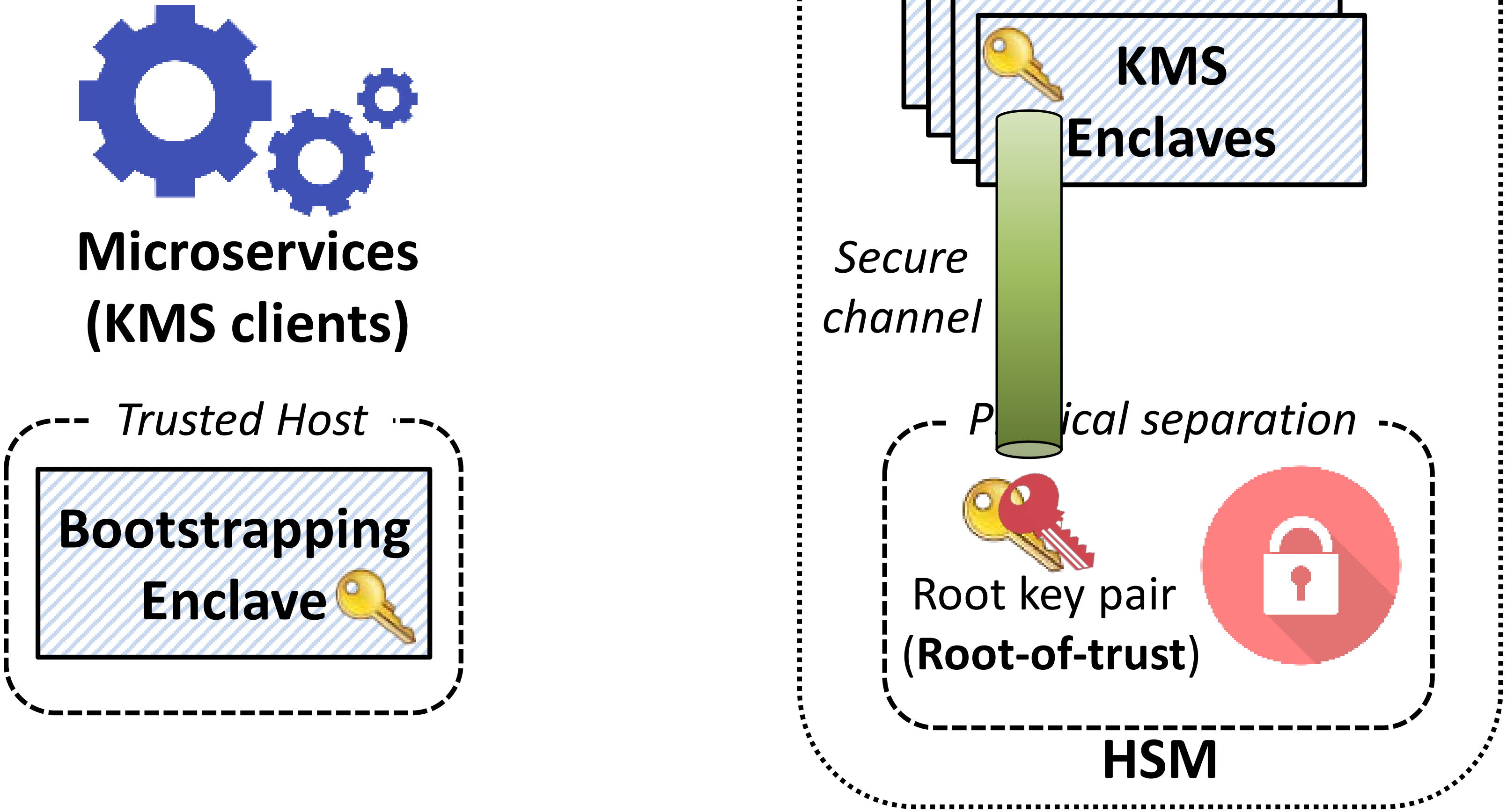
Secure bootstrapping

Secure bootstrapping ④ :
The bootstrapping enclave shares the public key



Secure bootstrapping

Secure bootstrapping ⑤ :
The KMS enclaves attest the HSM and build secure channels



Microservices (KMS clients)

Trusted Host

Bootstrapping Enclave

Untrusted Platform

KMS Enclaves

Secure channel

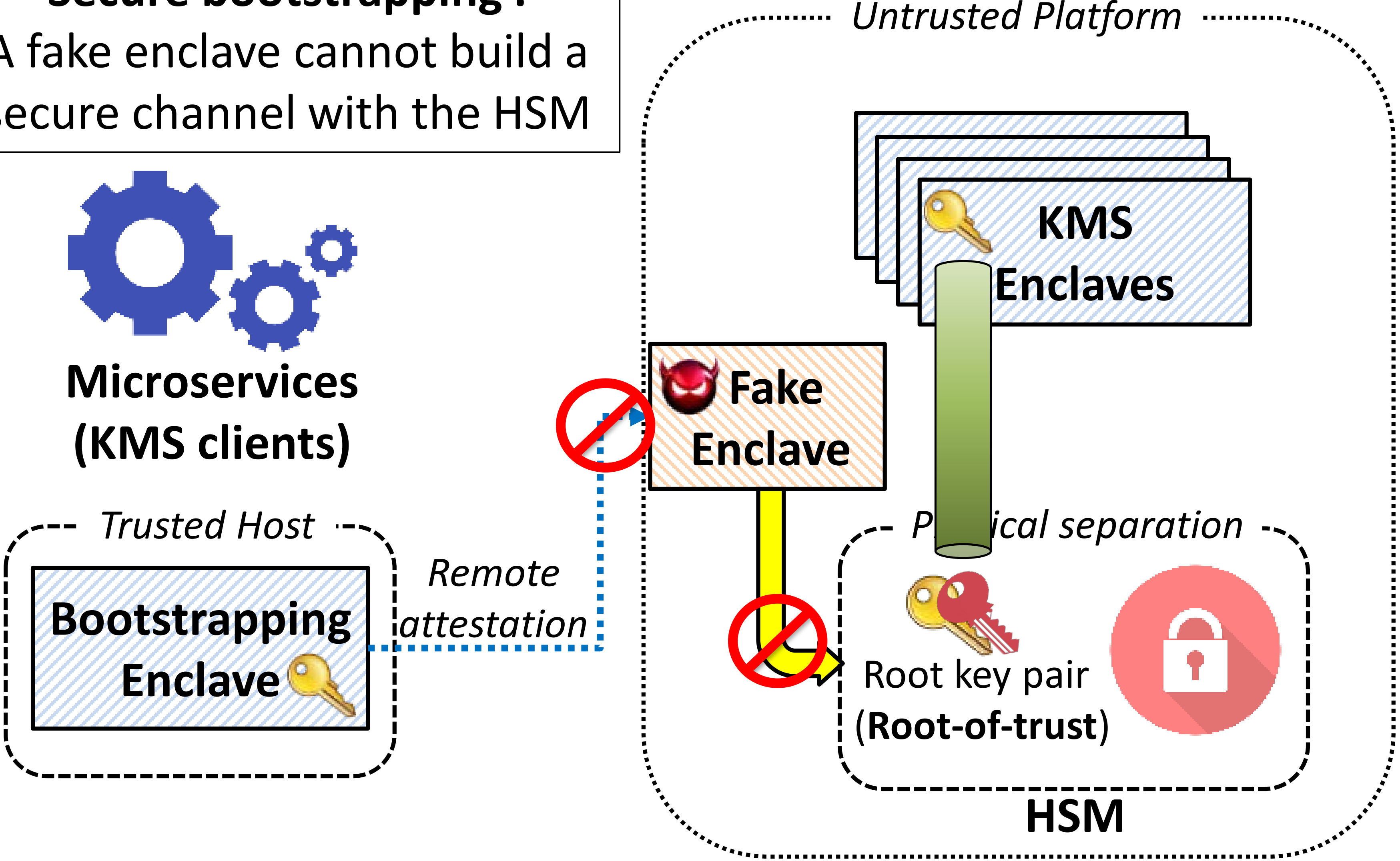
Physical separation

Root key pair **(Root-of-trust)**

HSM

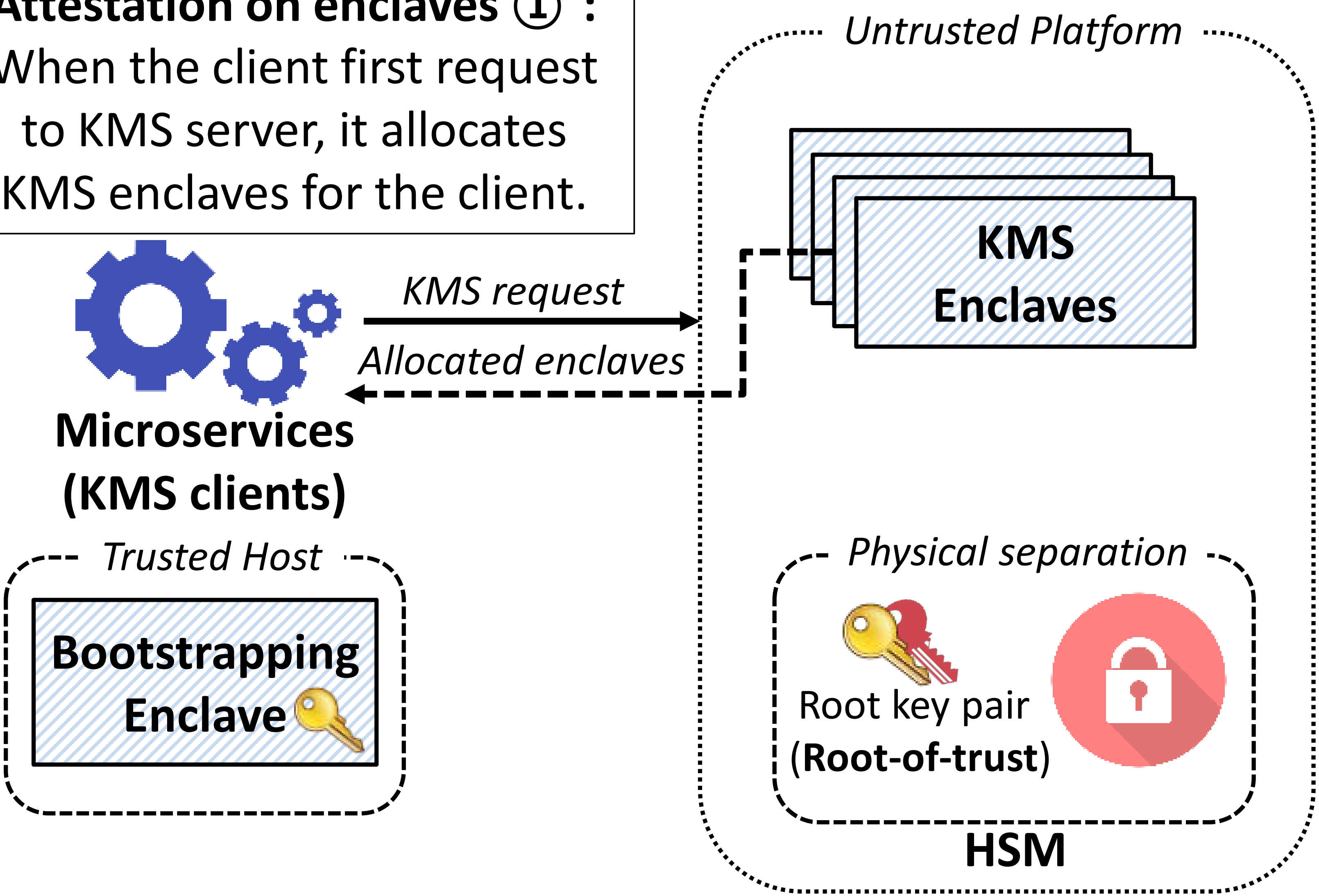
Secure bootstrapping

Secure bootstrapping :
A fake enclave cannot build a secure channel with the HSM



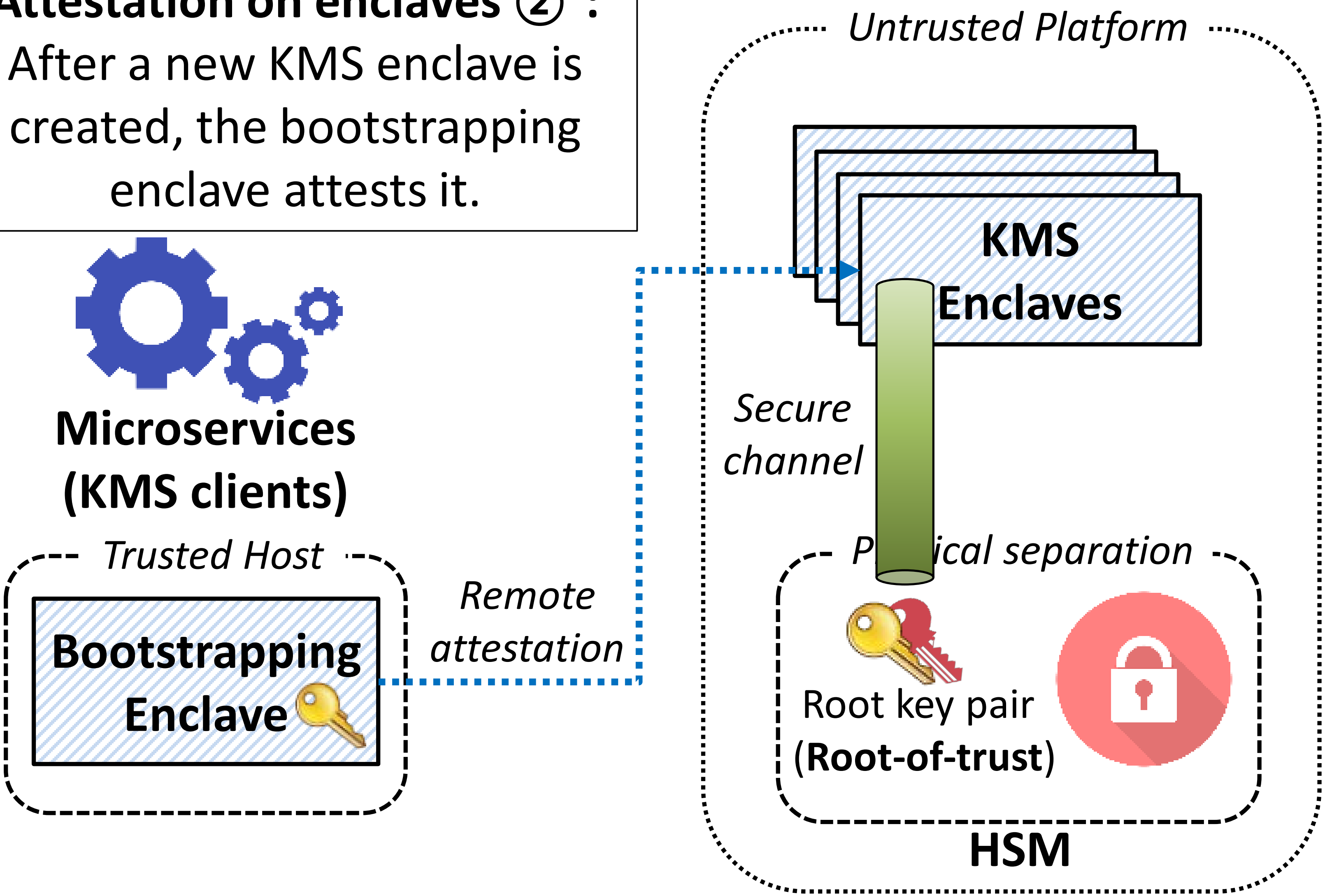
Attestation on SGX Instances

Attestation on enclaves ① : When the client first request to KMS server, it allocates KMS enclaves for the client.



Attestation on SGX Instances

Attestation on enclaves ② :
After a new KMS enclave is created, the bootstrapping enclave attests it.



Untrusted Platform

KMS Enclaves

Secure channel

P ical separation

Root key pair **(Root-of-trust)**

HSM

Microservices (KMS clients)

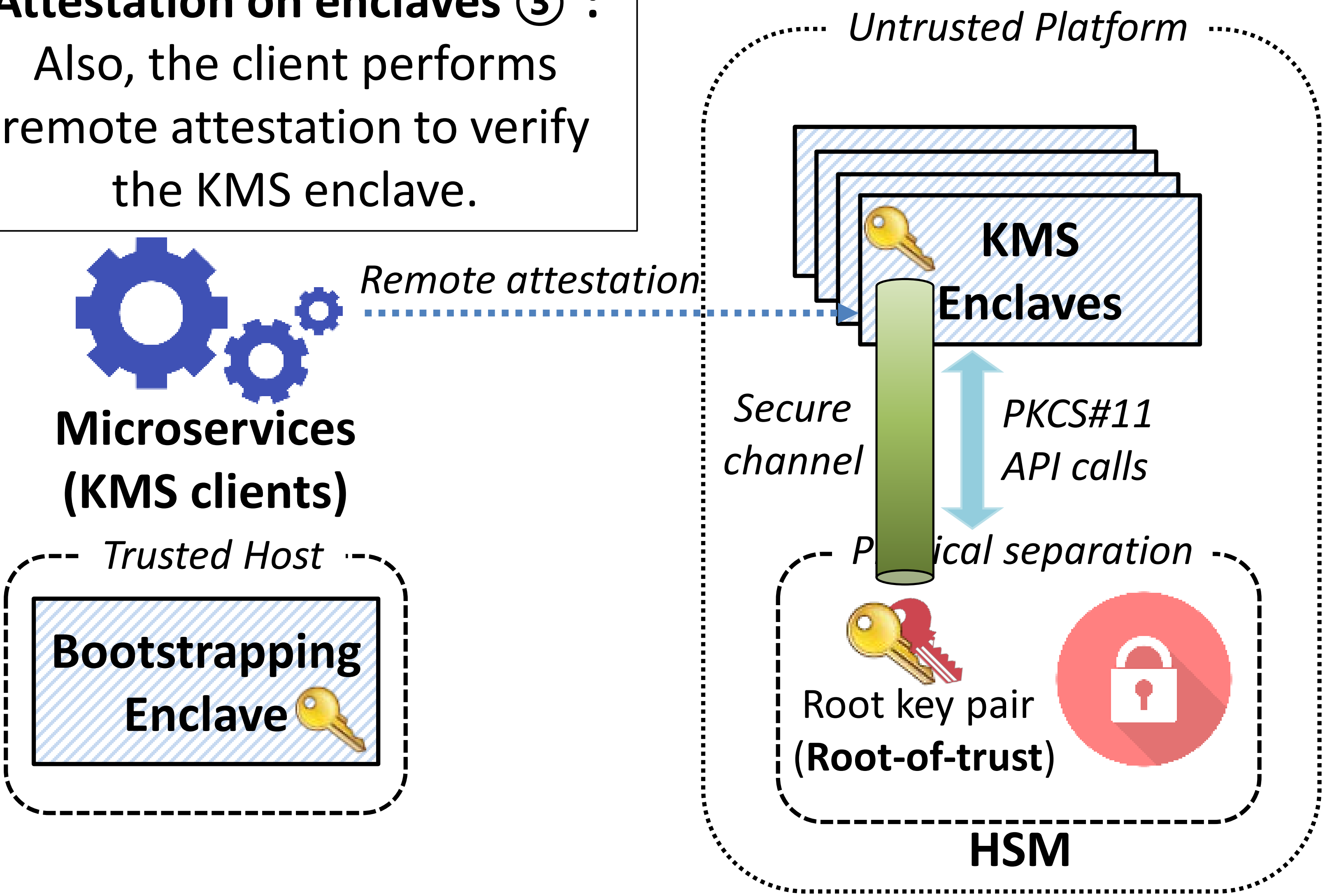
Trusted Host

Bootstrapping Enclave

Remote attestation

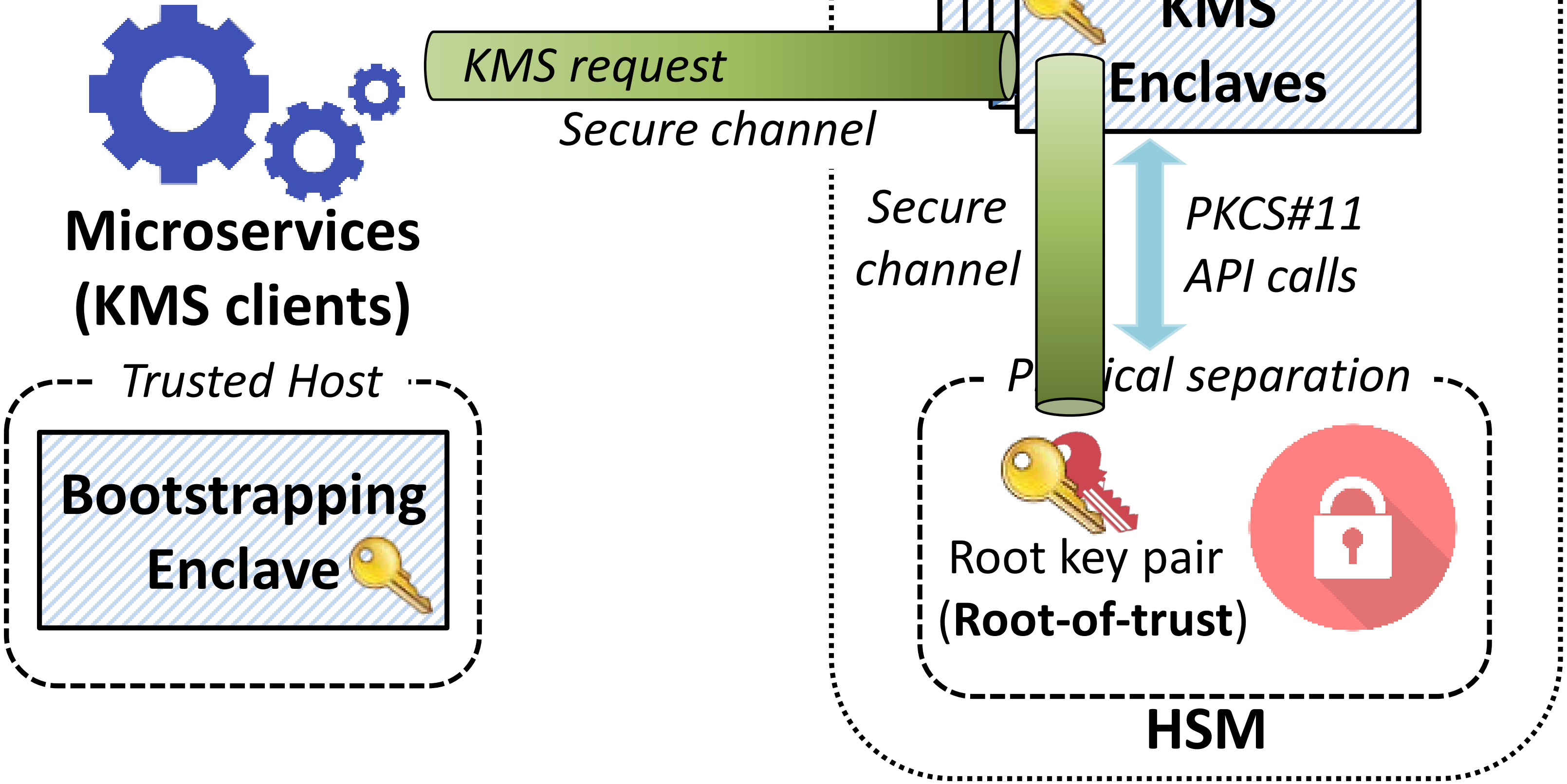
Attestation on SGX Instances

Attestation on enclaves ③ :
Also, the client performs remote attestation to verify the KMS enclave.



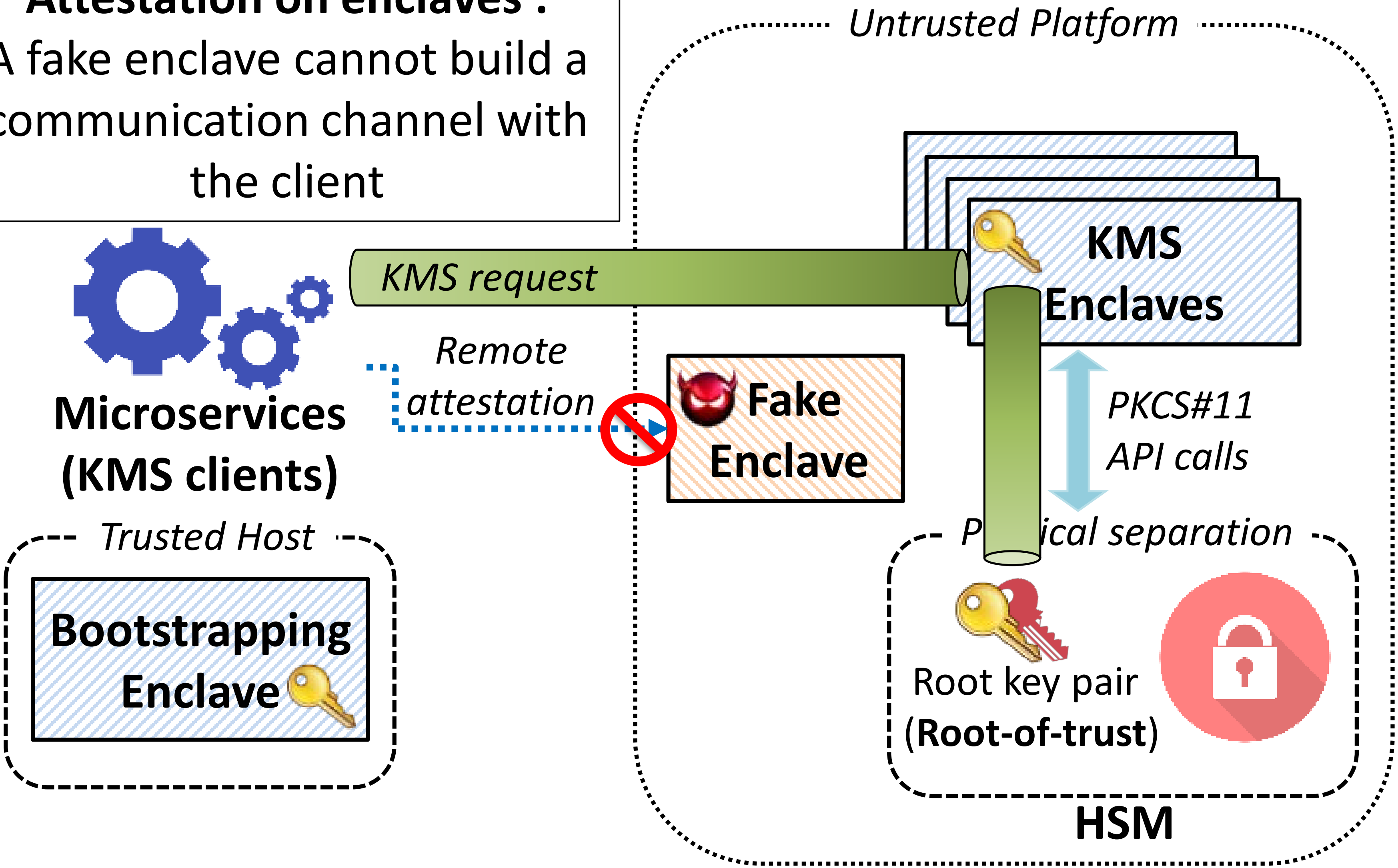
Attestation on SGX Instances

Attestation on enclaves ④ :
After the remote attestation, the client sends encrypted KMS requests to the enclave

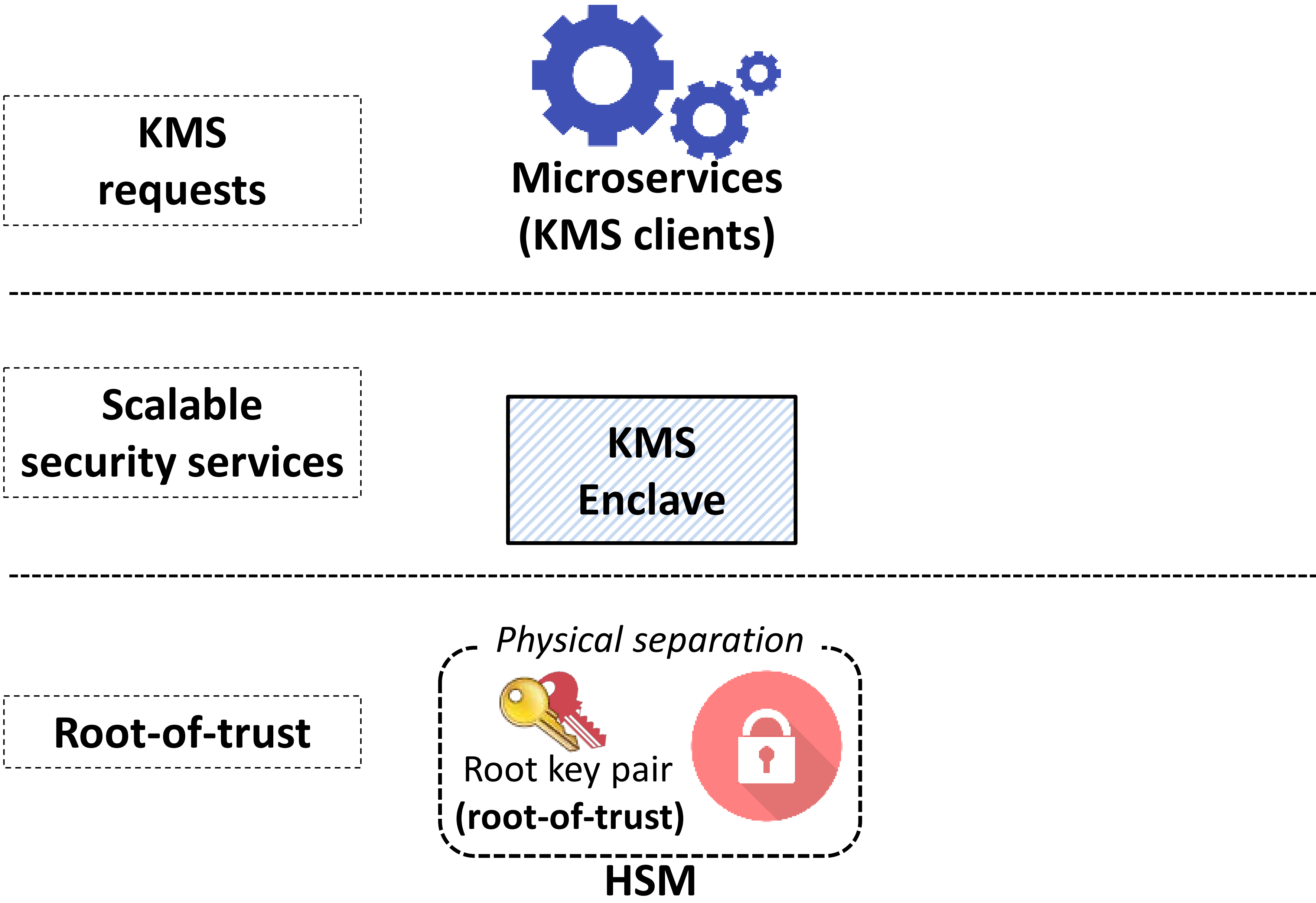


Attestation on SGX Instances

Attestation on enclaves :
A fake enclave cannot build a communication channel with the client



Hierarchical Design for Scaling



KMS requests

Microservices (KMS clients)

Scalable security services

KMS Enclave

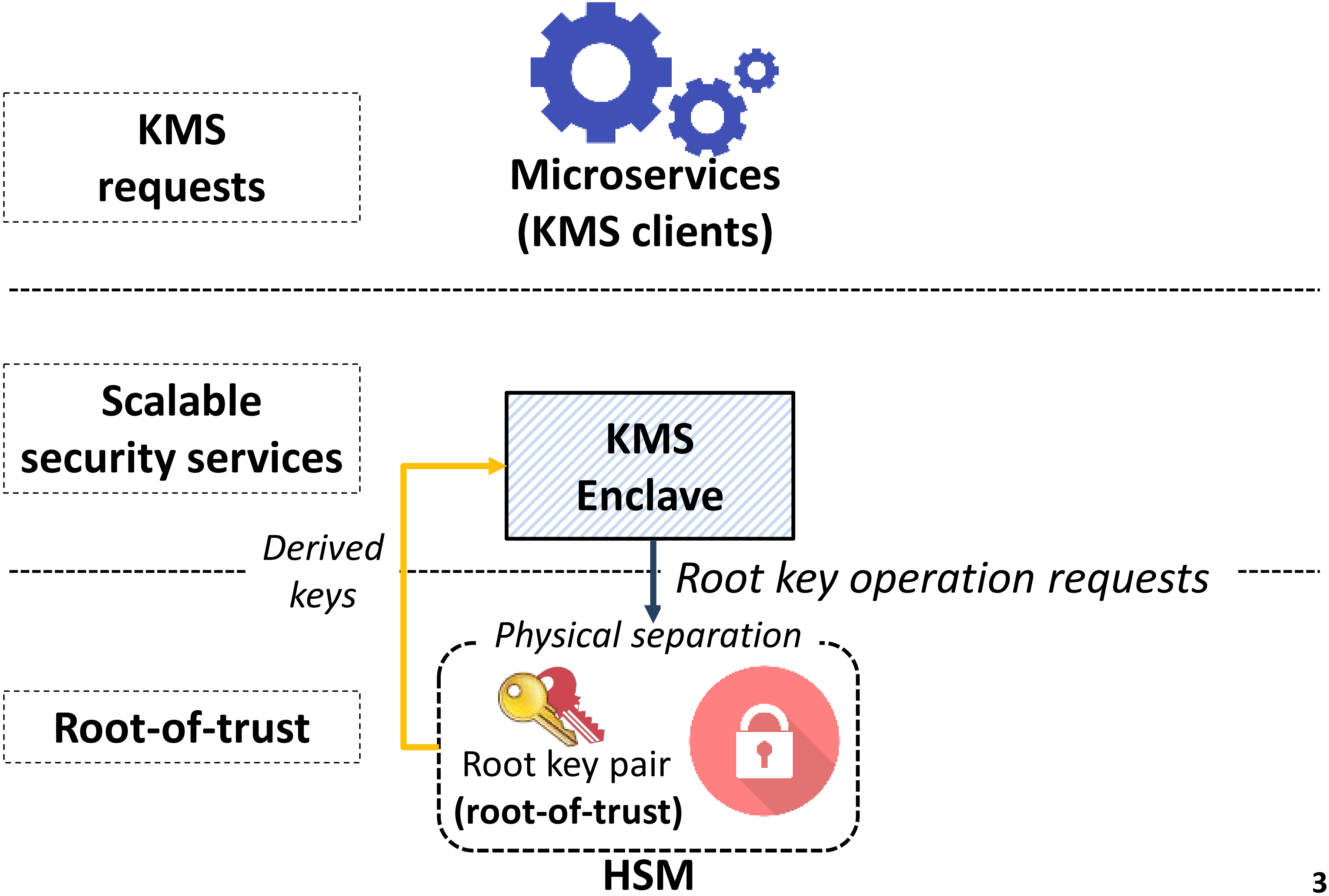
Root-of-trust

Physical separation

Root key pair **(root-of-trust)**

HSM

Hierarchical Design for Scaling



KMS requests

Microservices (KMS clients)

Scalable security services

KMS Enclave

Derived keys

Root key operation requests

Physical separation

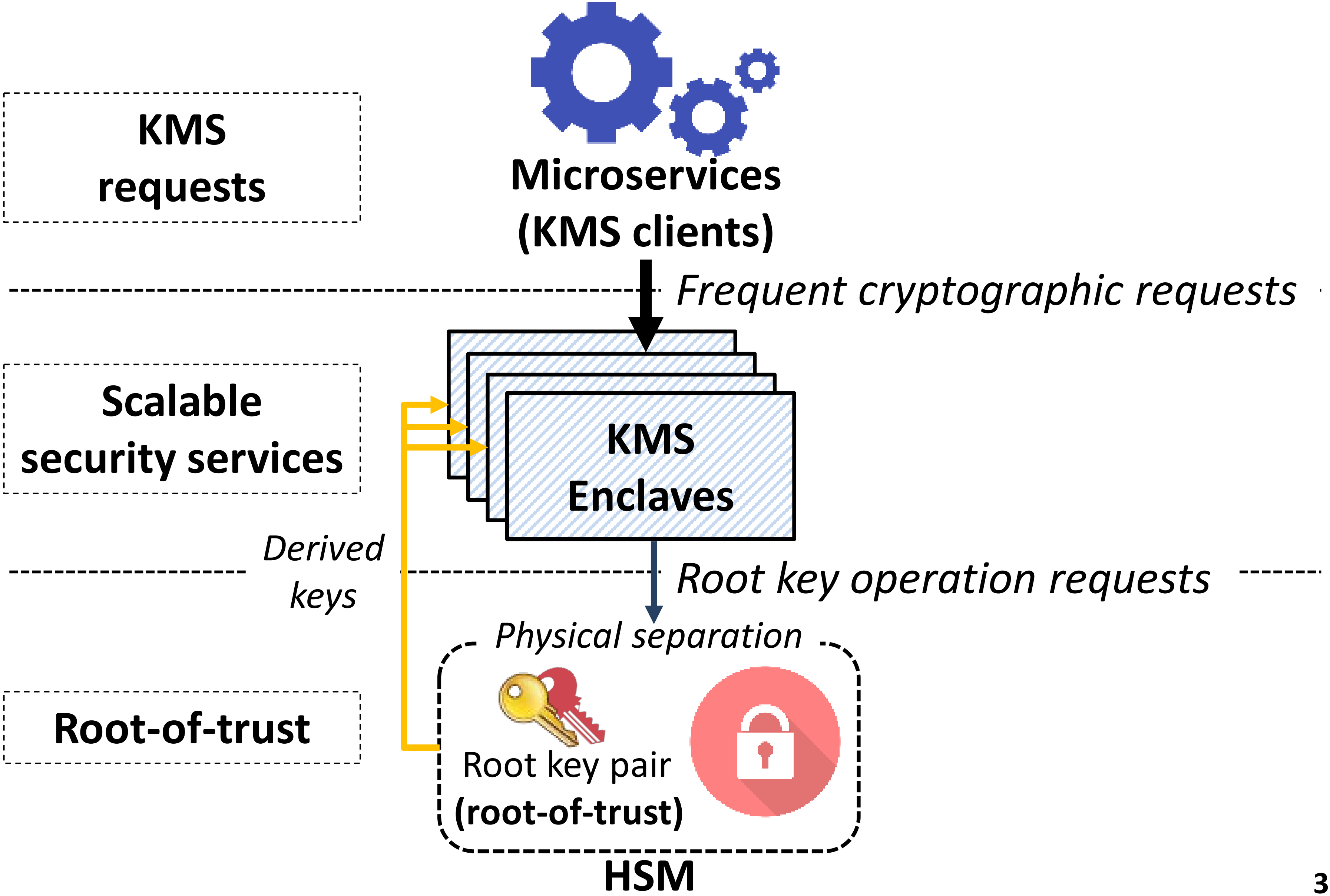
Root key pair

(root-of-trust)

HSM

Root-of-trust

Hierarchical Design for Scaling



KMS requests

Scalable security services

Root-of-trust

Microservices (KMS clients)

Frequent cryptographic requests

KMS Enclaves

Derived keys

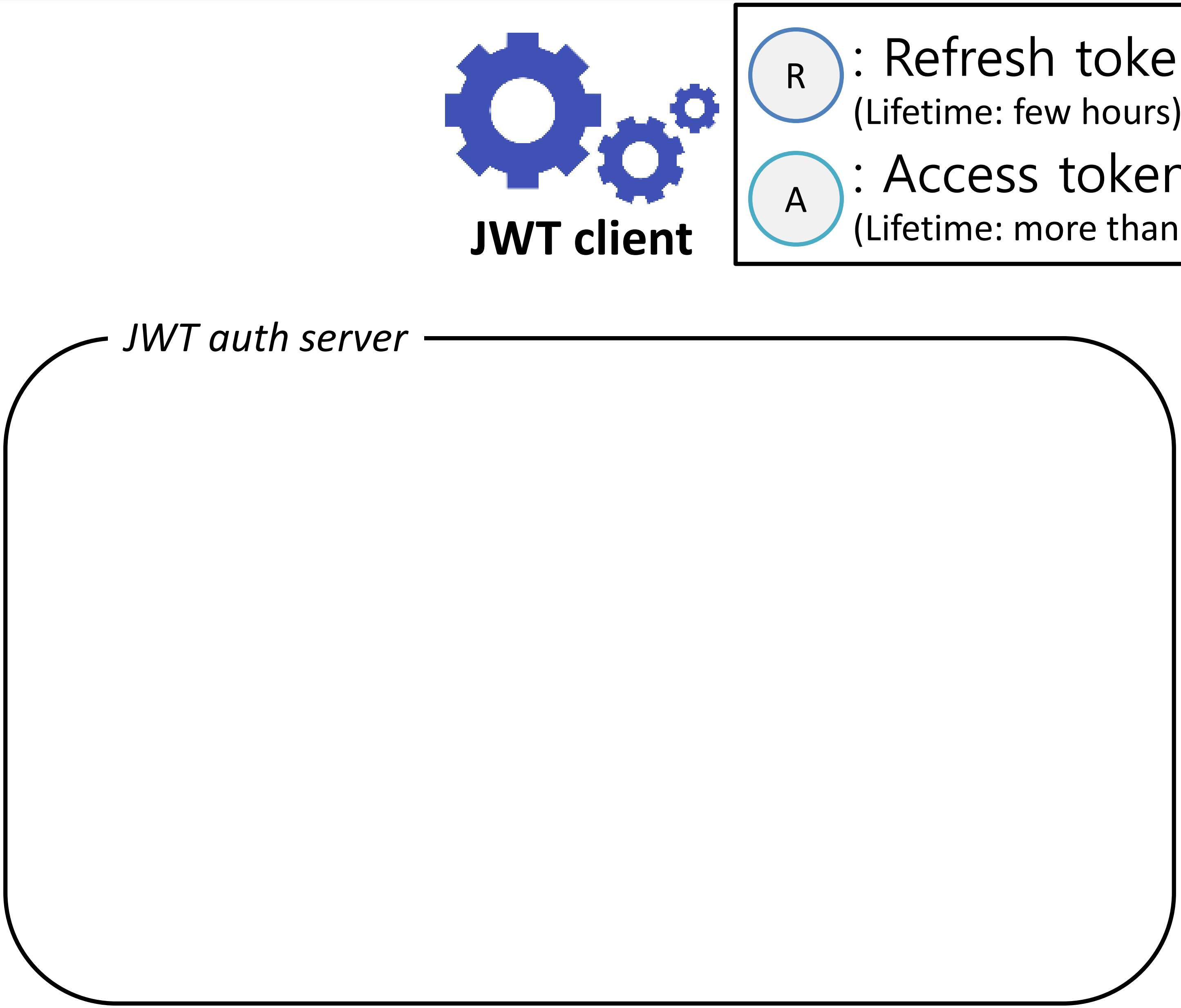
Root key operation requests

Physical separation

Root key pair **(root-of-trust)**

HSM

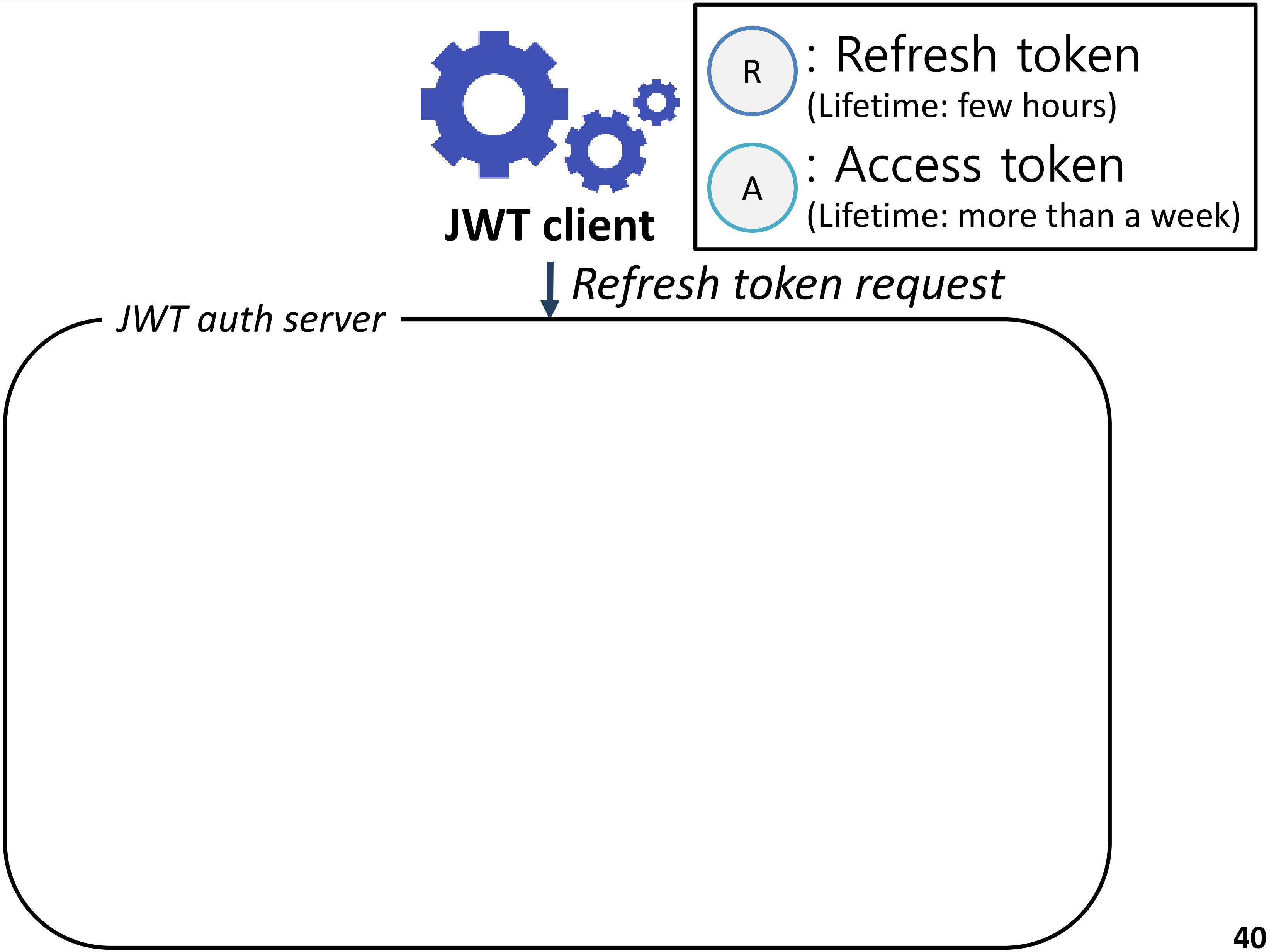
JSON Web Token (JWT) for Microservice



R : Refresh token
(Lifetime: few hours)

A : Access token
(Lifetime: more than a week)

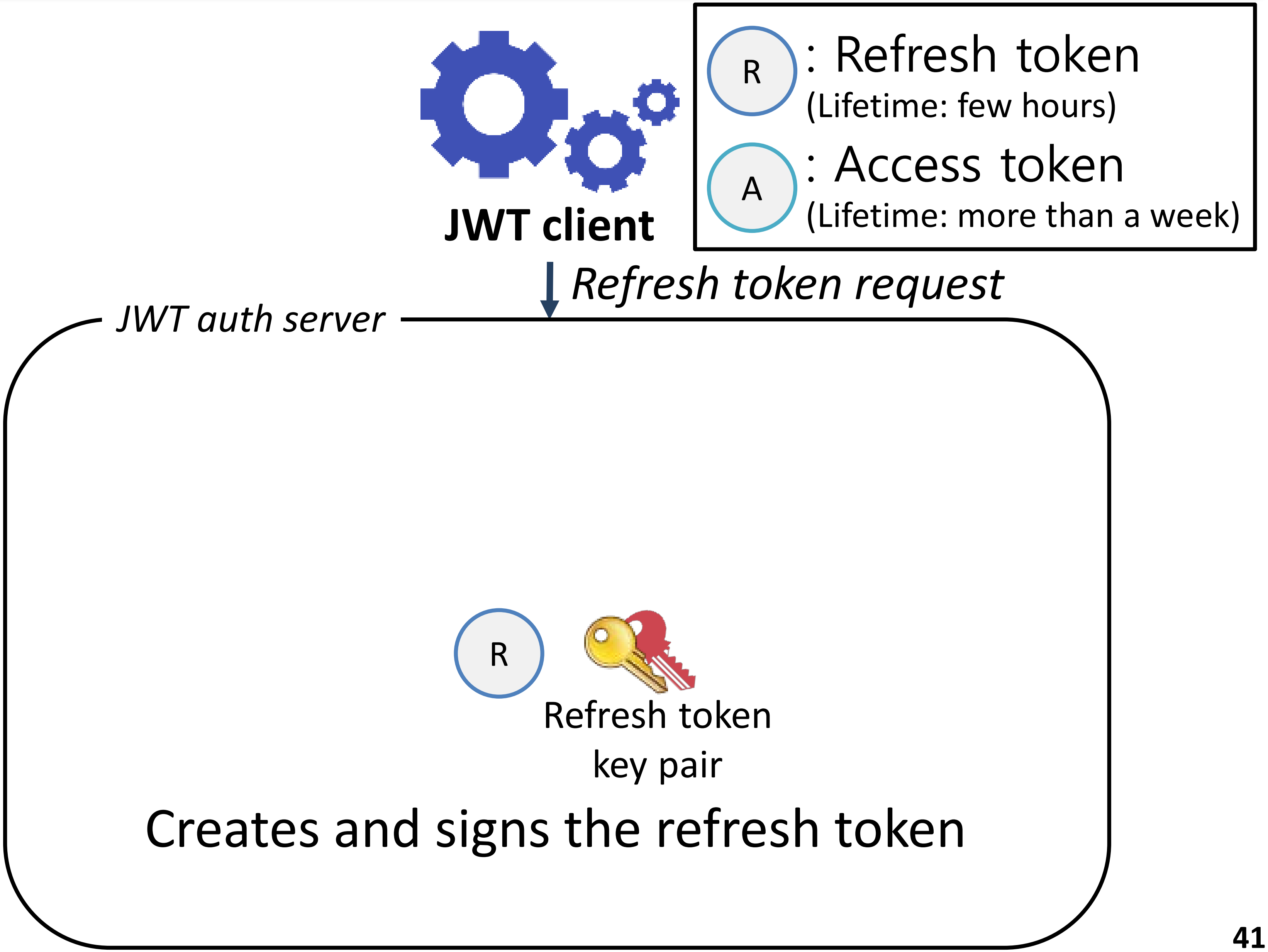
JSON Web Token (JWT) for Microservice



R : Refresh token
(Lifetime: few hours)

A : Access token
(Lifetime: more than a week)

JSON Web Token (JWT) for Microservice



JWT client

R : Refresh token (Lifetime: few hours)

A : Access token (Lifetime: more than a week)

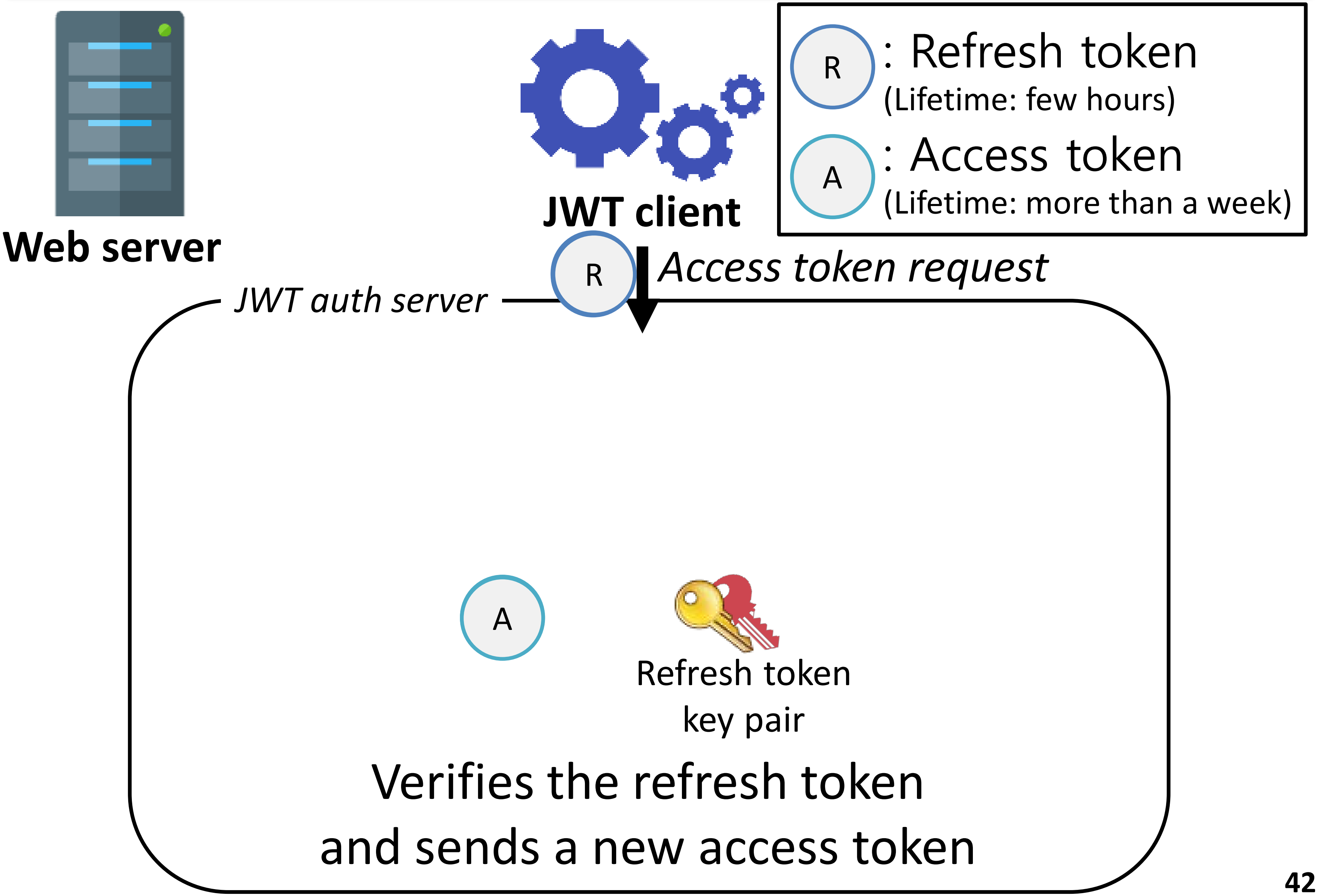
Refresh token request

JWT auth server

Refresh token key pair

Creates and signs the refresh token

JSON Web Token (JWT) for Microservice



: Refresh token (Lifetime: few hours)

: Access token (Lifetime: more than a week)

Web server

JWT client

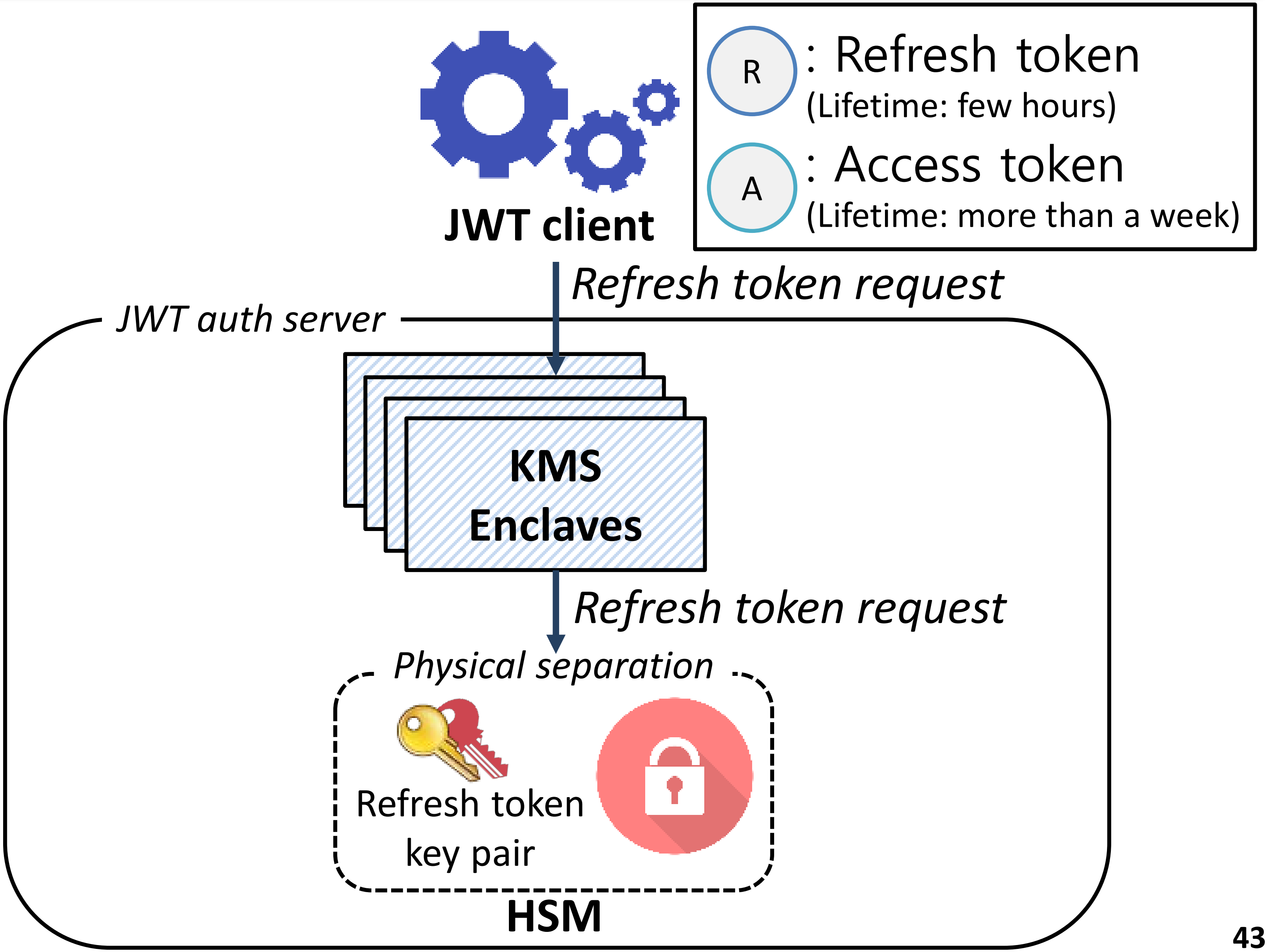
Access token request

JWT auth server

Refresh token key pair

Verifies the refresh token and sends a new access token

Application Case Study : JWT Management



R : Refresh token (Lifetime: few hours)

A : Access token (Lifetime: more than a week)

JWT client

Refresh token request

JWT auth server

KMS Enclaves

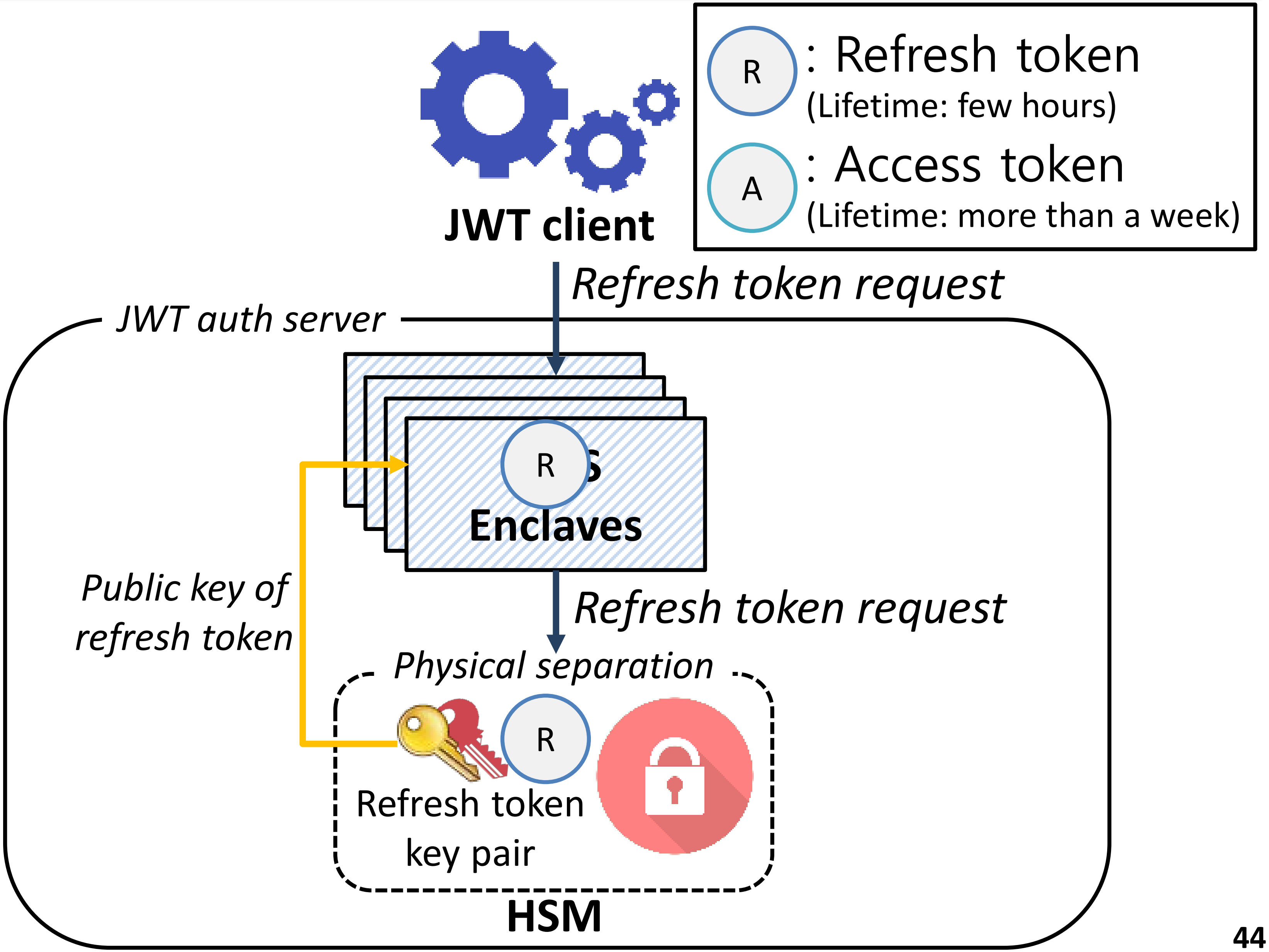
Refresh token request

Physical separation

Refresh token key pair

HSM

Application Case Study : JWT Management



R : Refresh token (Lifetime: few hours)

A : Access token (Lifetime: more than a week)

JWT client

Refresh token request

JWT auth server

R

S

Enclaves

Public key of refresh token

Refresh token request

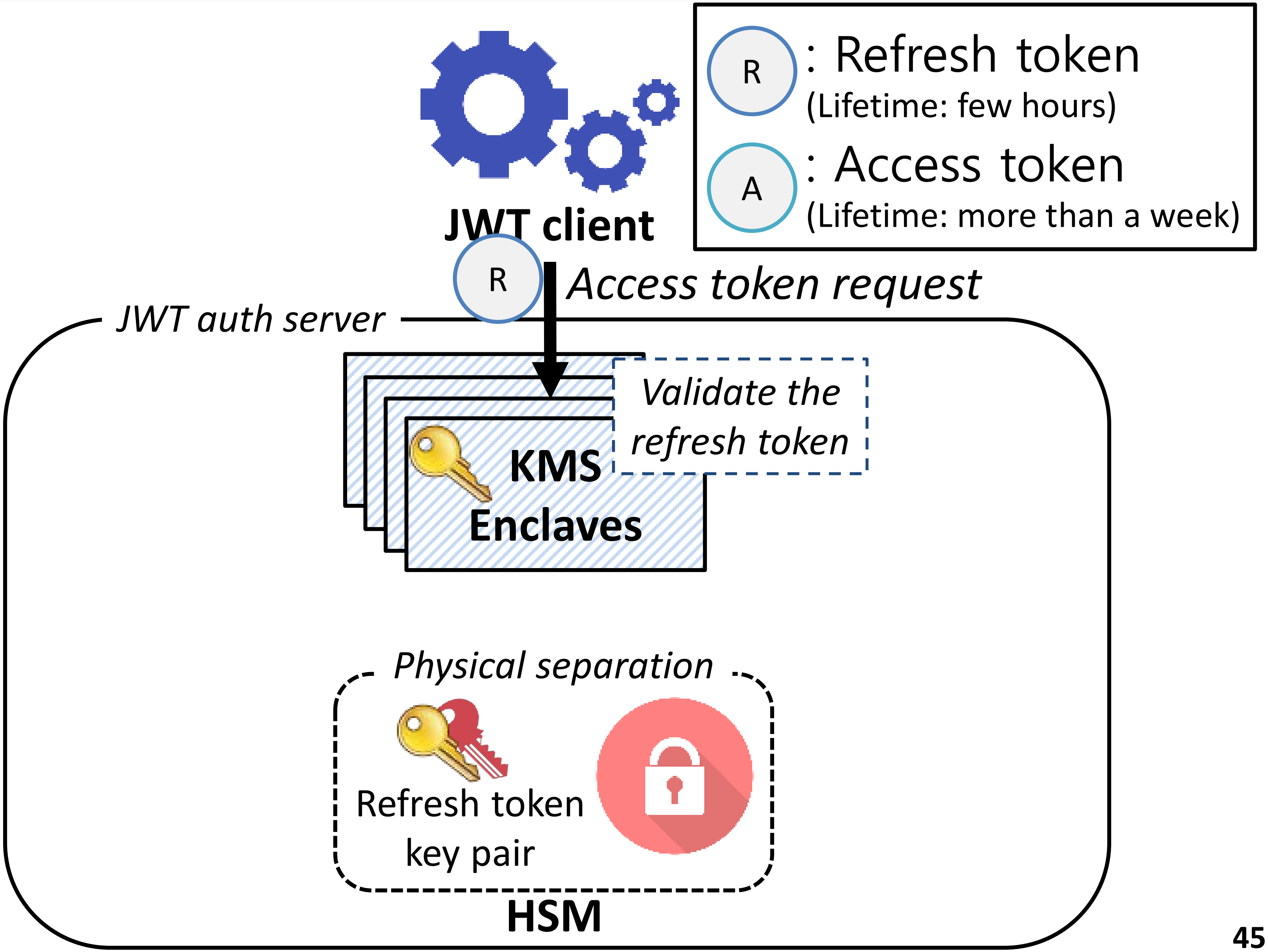
Physical separation

R

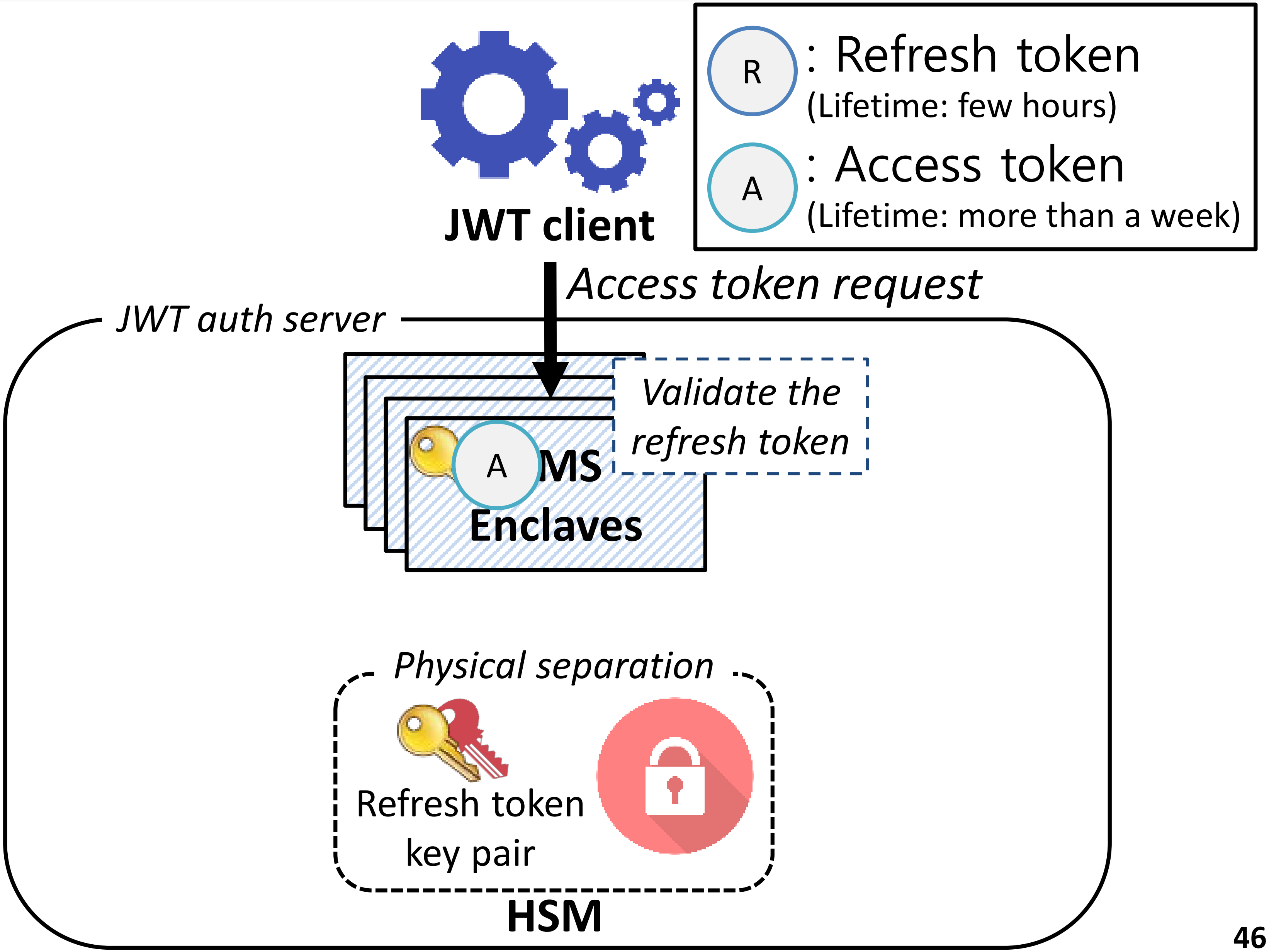
Refresh token key pair

HSM

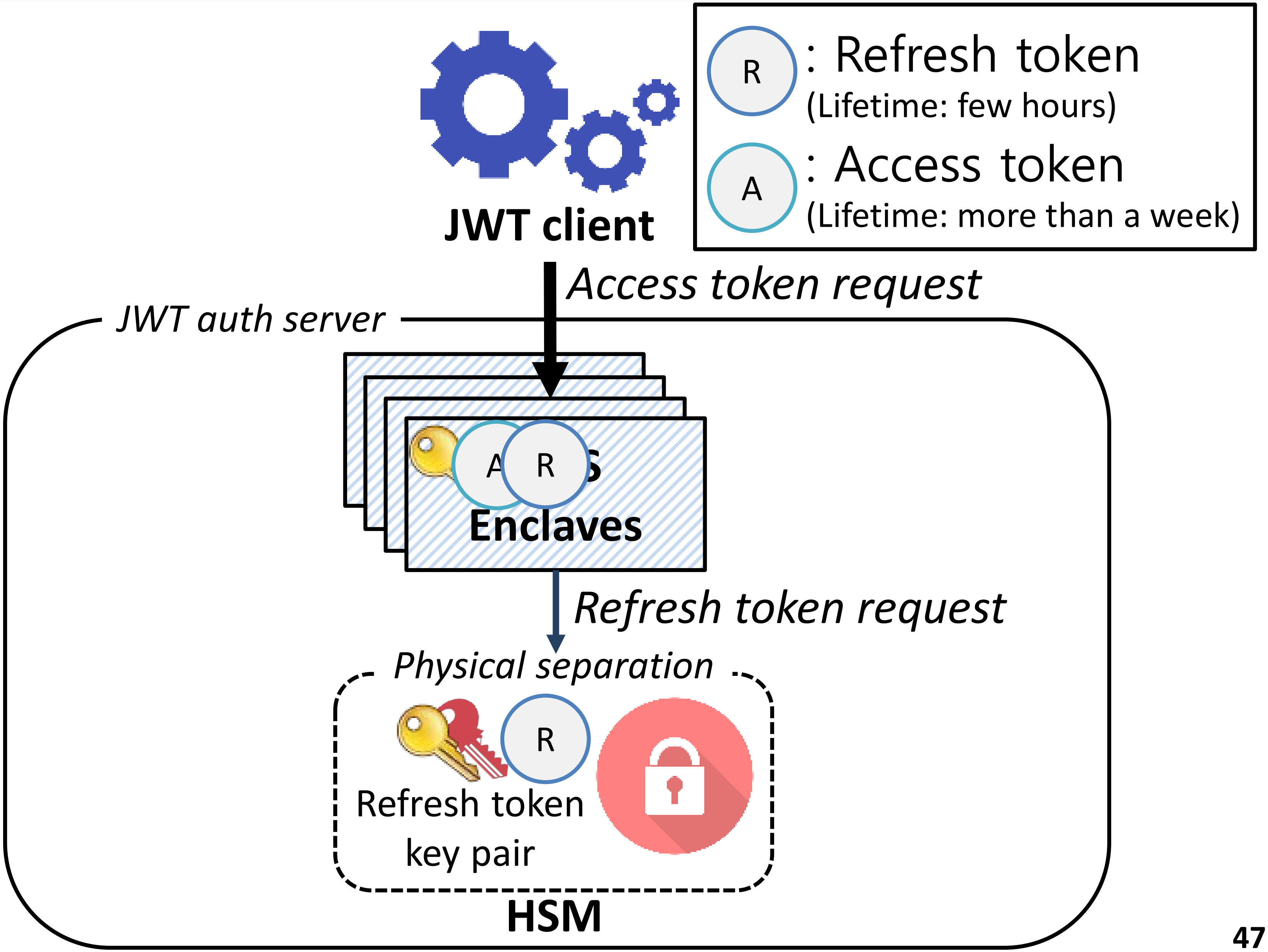
Application Case Study : JWT Management



Application Case Study : JWT Management



Application Case Study : JWT Management



R : Refresh token (Lifetime: few hours)

A : Access token (Lifetime: more than a week)

JWT client

Access token request

JWT auth server

A R

Enclaves

Refresh token request

Physical separation

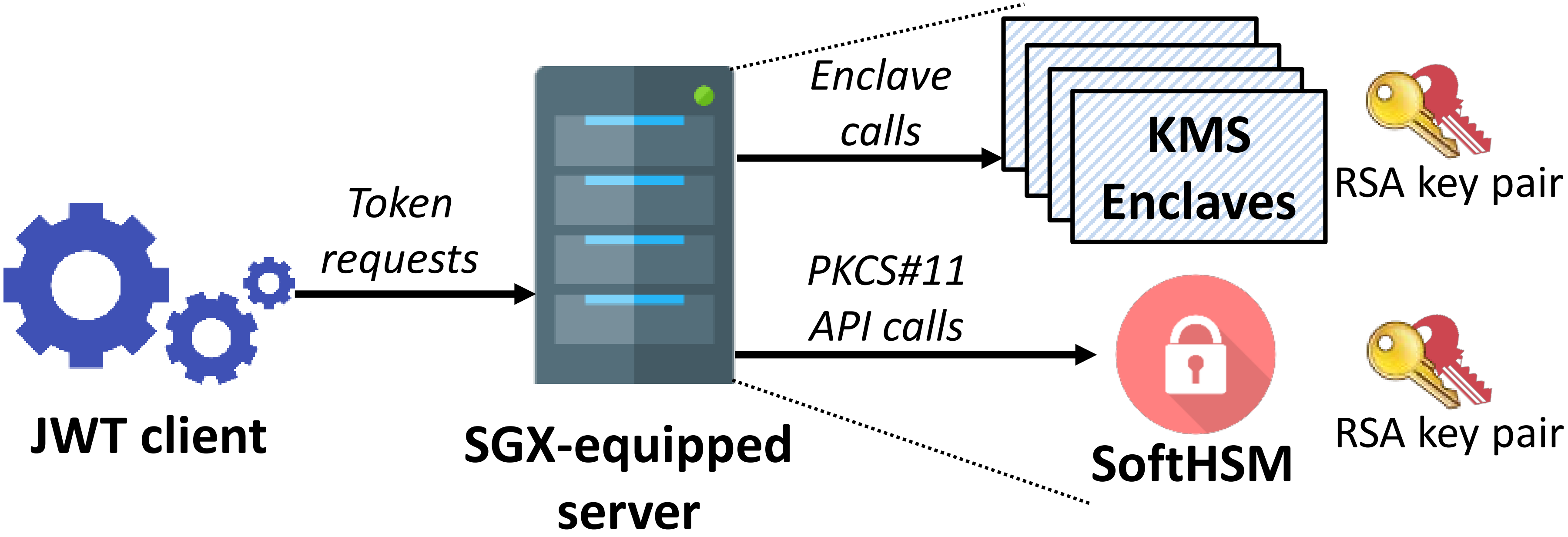
R

Refresh token key pair

HSM

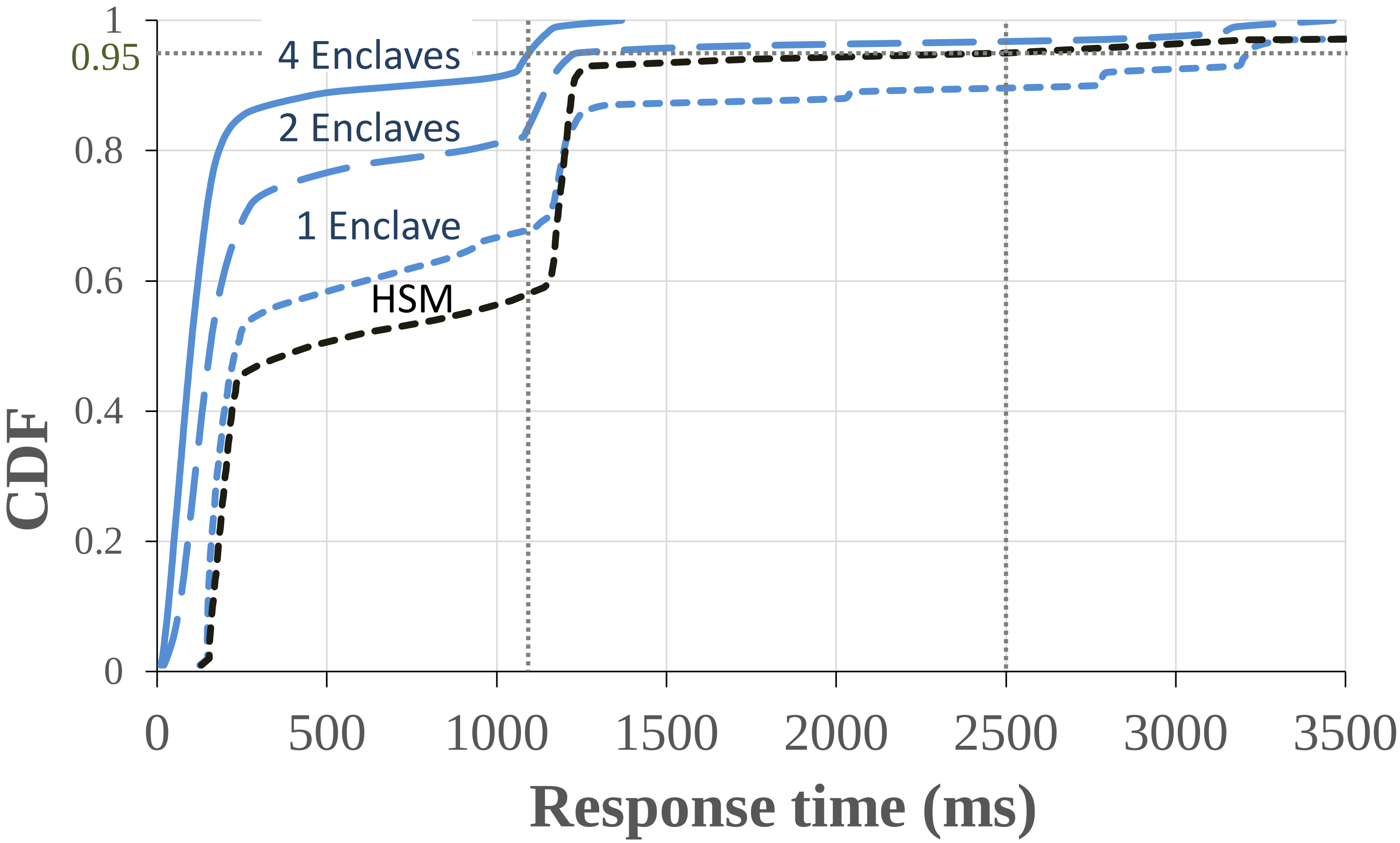
Preliminary Evaluation

- **Environment setup**
  - CPU: Quad-core Intel Xeon E3-1280 v6 (SGX-enabled)
  - Intel SGX Linux SDK version 2.5
  - We use SoftHSM to emulate an HSM device.
  - Each enclave and HSM performs the same SHA-256 with RSA-2048 signing



Preliminary Evaluation: Latency Improvement

- **Scaling out KMS enclaves for latency improvement**



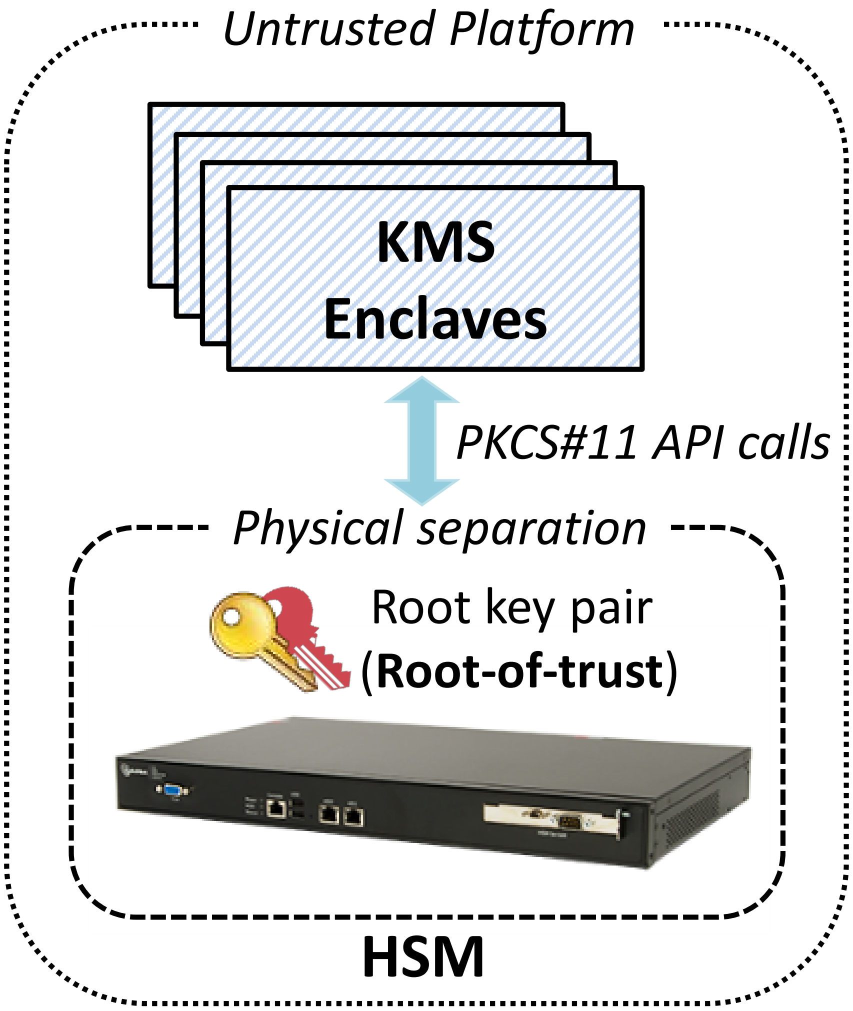
Preliminary Evaluation: Cost-effective Scaling

Approach for KMS	Equipment	Performance (RSA-2048 sign)	Price	tps/$
ScaleTrust (on-premises SGX machine)	Xeon E3-1280 v6 CPU (Quad, 4.2 GHz)	3,600 tps	$500	**7.2**
On-premises HSMs-only	Luna SA A790 HSM	10,000 tps	$29,900	**0.33**
ScaleTrust (in Azure cloud)	Xeon E-2176G CPU (Quad, 4.7 GHz)	> 3,600 tps (estimated)	$500 per month	**> 7.2 for a month**
Cloud HSM (Azure HSM)	Luna SA A790 HSM	10,000 tps	$5000 + $3,541 per month	**1.17 for a month**

*tps = transactions per second

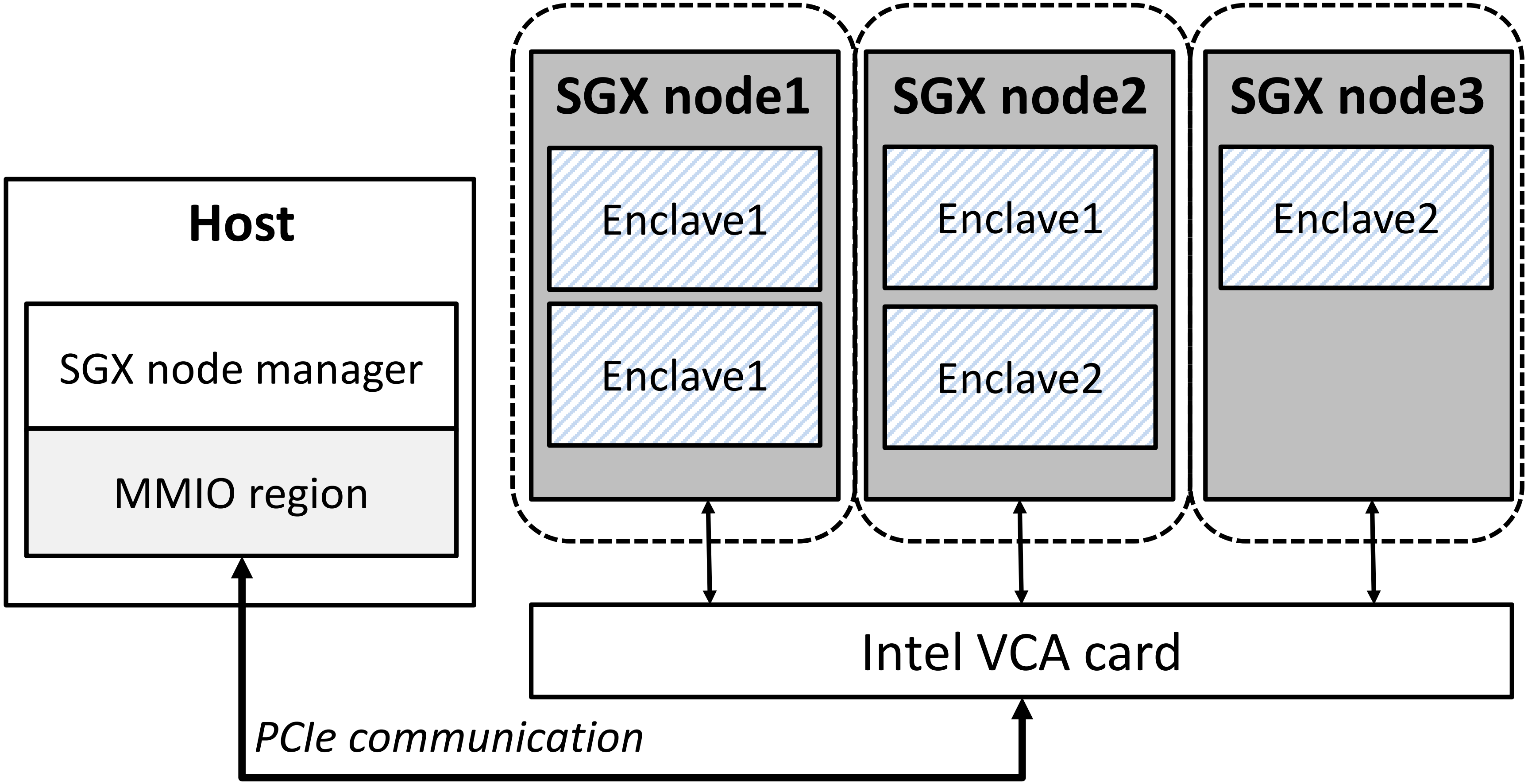
Future work

- **Evaluation with a real HSM device**



Future work

- **Physical separation by Intel VCA (SGX card)**



Conclusion

- We explore new design space to address the limited **scalability of HSMs** by combining TEE technology
- ScaleTrust preserves **chain-of-trust** from an HSM to clients
- ScaleTrust utilizes HSMs and SGX enclaves in a hierarchical model to **relieve the burden of HSMs**
- Our JWT case study shows that ScaleTrust can be applied to key management for microservices.

Thank You