
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

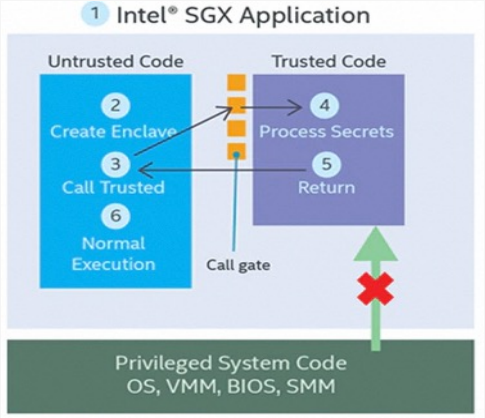
CoCo-AS: First Confidential Computing Attestation Solution of CNCF

Jiale Zhang - Alibaba Cloud
Dave Chen – ARM China

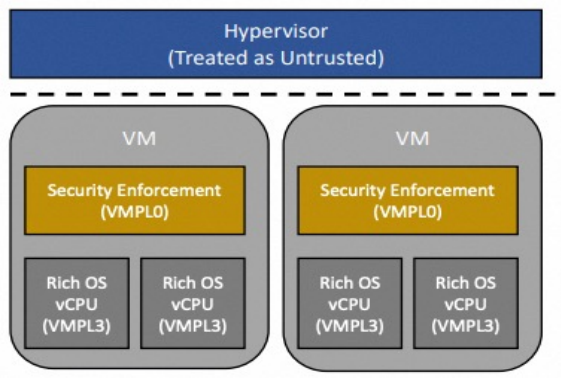
Introduction of Confidential Computing

Confidential computing technology is based on hardware to provide a Trusted Execution Environment (TEE), achieving confidentiality, integrity, and security protection for "running" data.

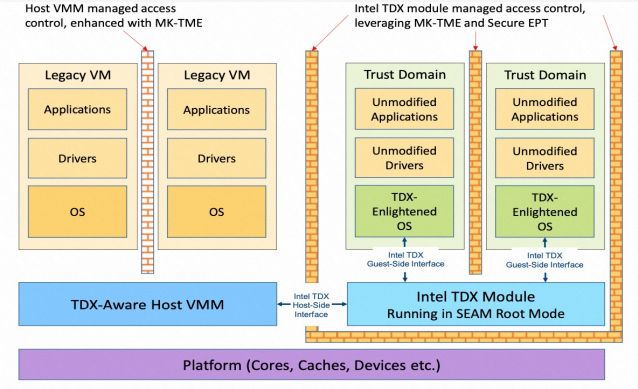
Hardware based implementation: memory encryption, memory isolation, CPU state protection, Attestation.



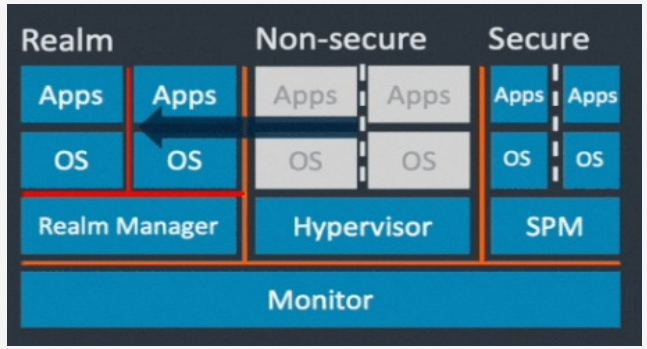
Intel SGX【1】



AMD SEV-SNP【2】



Intel TDX【3】



ARM CCA【4】

【1】Intel Software Guard Extension：https://www.intel.com/content/www/us/en/develop/articles/intel-software-guard-extensions-tutorial-part-1-foundation.html
【2】AMD https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf
【3】Intel Trust Domain Extension： https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html
【4】Arm Confidential compute ：https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture

Confidential Computing Attestation

Key issue of confidential computing technology is how to prove to users the authenticity of the Trusted Execution Environment itself and the security capabilities provided by the Trusted Execution Environment are in line with expectations

Remote ATtestation procedureS (RATS) Architecture [RFC 9334]

- defines the abstract architecture of entities and the process of generating, transmitting, and evaluating evidence.
- Provides an understanding of the Attestation architecture, declaration content, and protocol

TEE hardware manufacturer support:

Intel: Architecture Enclave, PCK Cryptography System

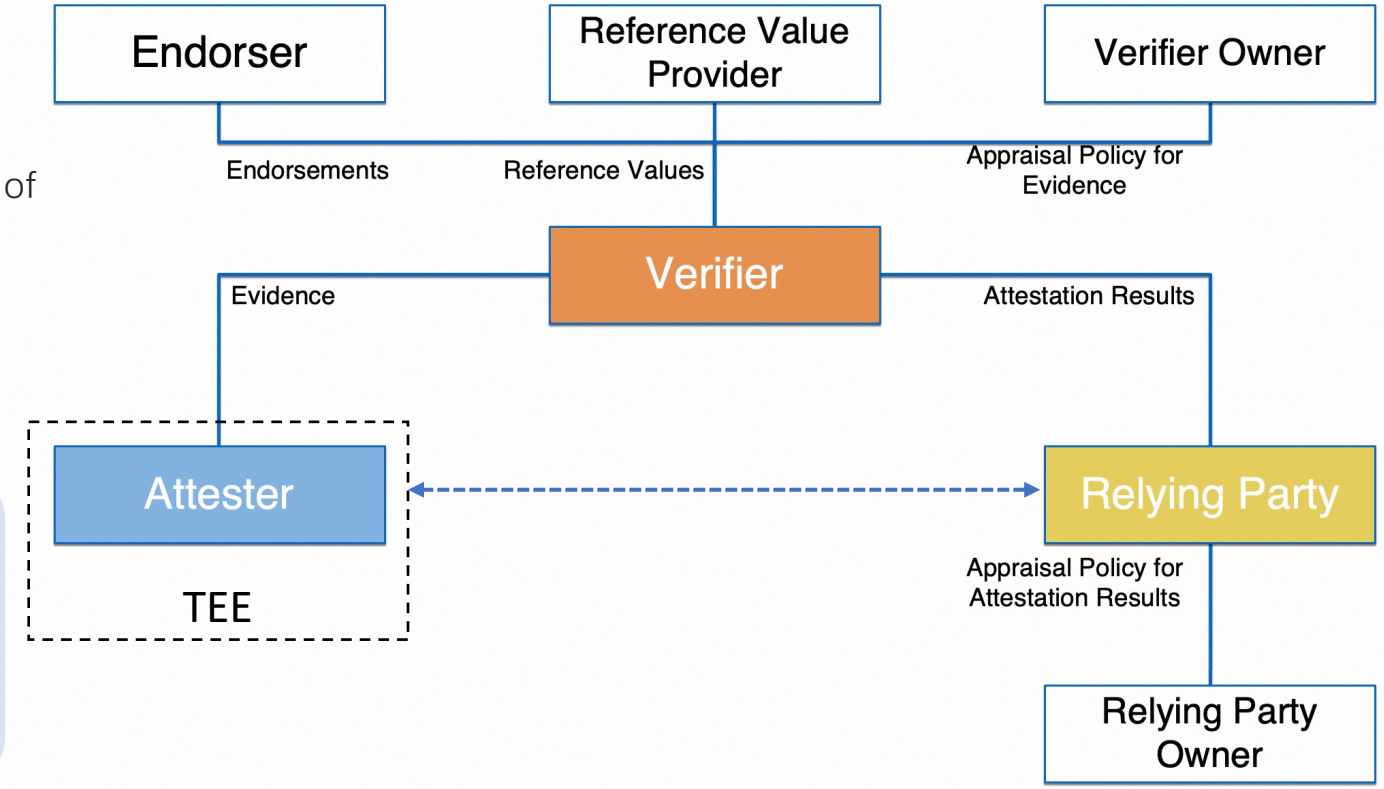
AMD: Certification report, certificate chain

ARM: Veraison

+

Software Architecture:

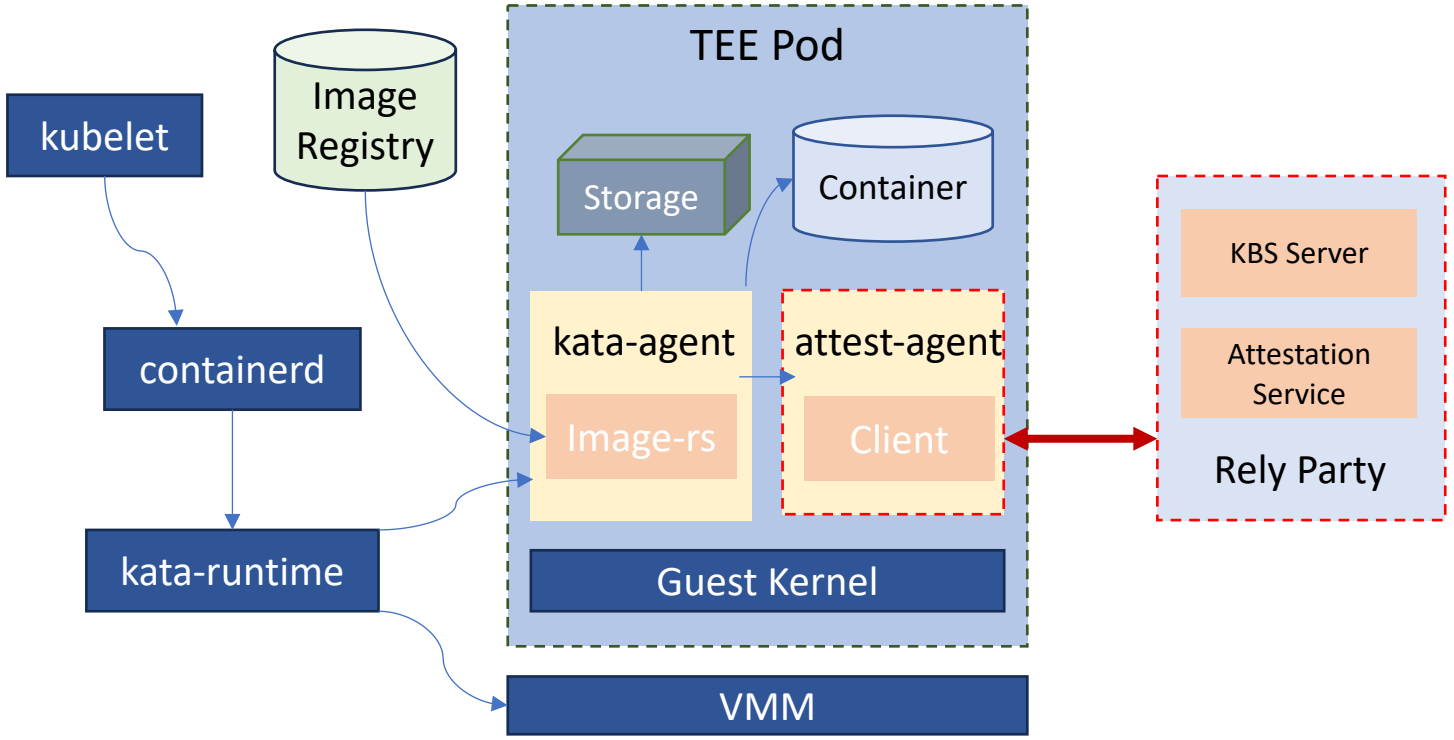
- Universal
- Flexible/adaptable



Confidential-Containaers (CoCo)

CONFIDENTIAL CONTAINERS

- Created in 2021, now is a CNCF sandbox project
- Committed to enable cloud native confidential computing by leveraging Trusted Execution Environment to protect containers and data.
- Active contributors, currently released version v0.8.0



features:

- Pod runs in TEE with confidentiality protection
- Special kata-runtime, VMM, Kernel
- Container images are protected by encryption and signature
- Image management system inside Pod
- **Attestation System**

CoCo Attestation System (CoCo-AS)

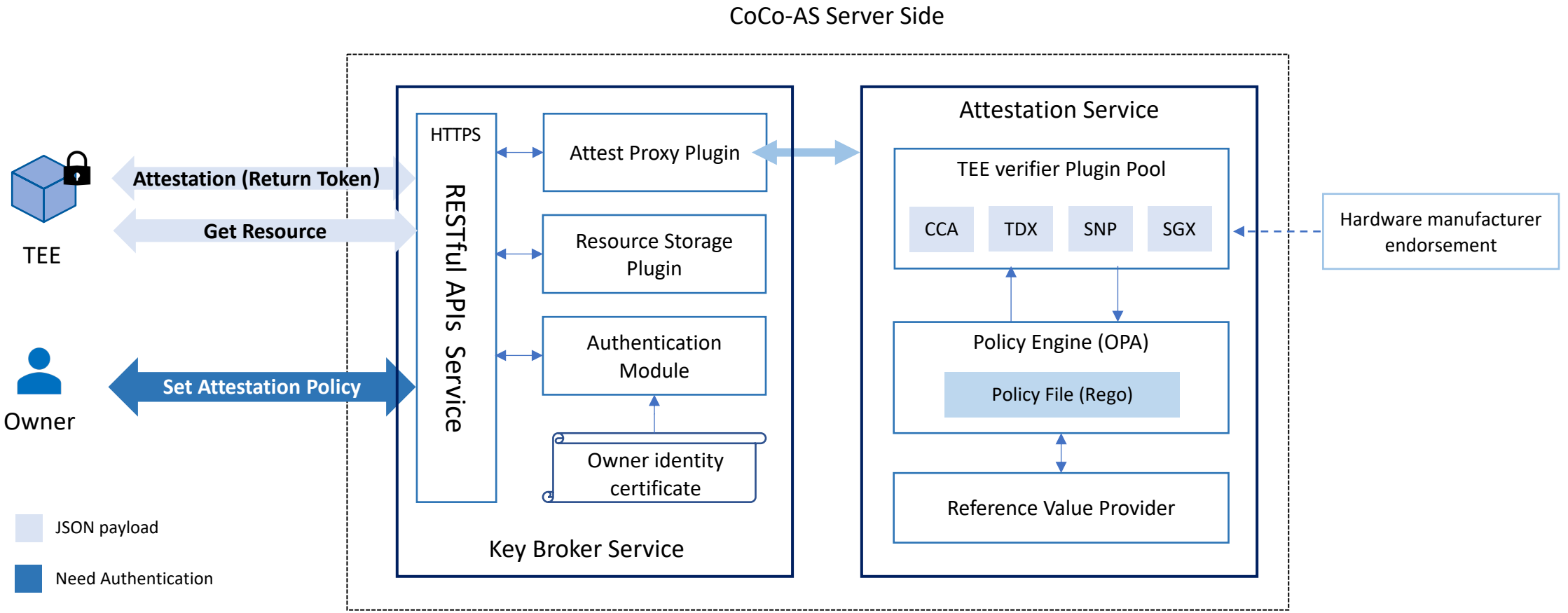
Attestation System:

A unified Attestation software solution that supports multi platform TEE and can be applied to a wide range of cloud native confidential computing scenarios, including CoCo.

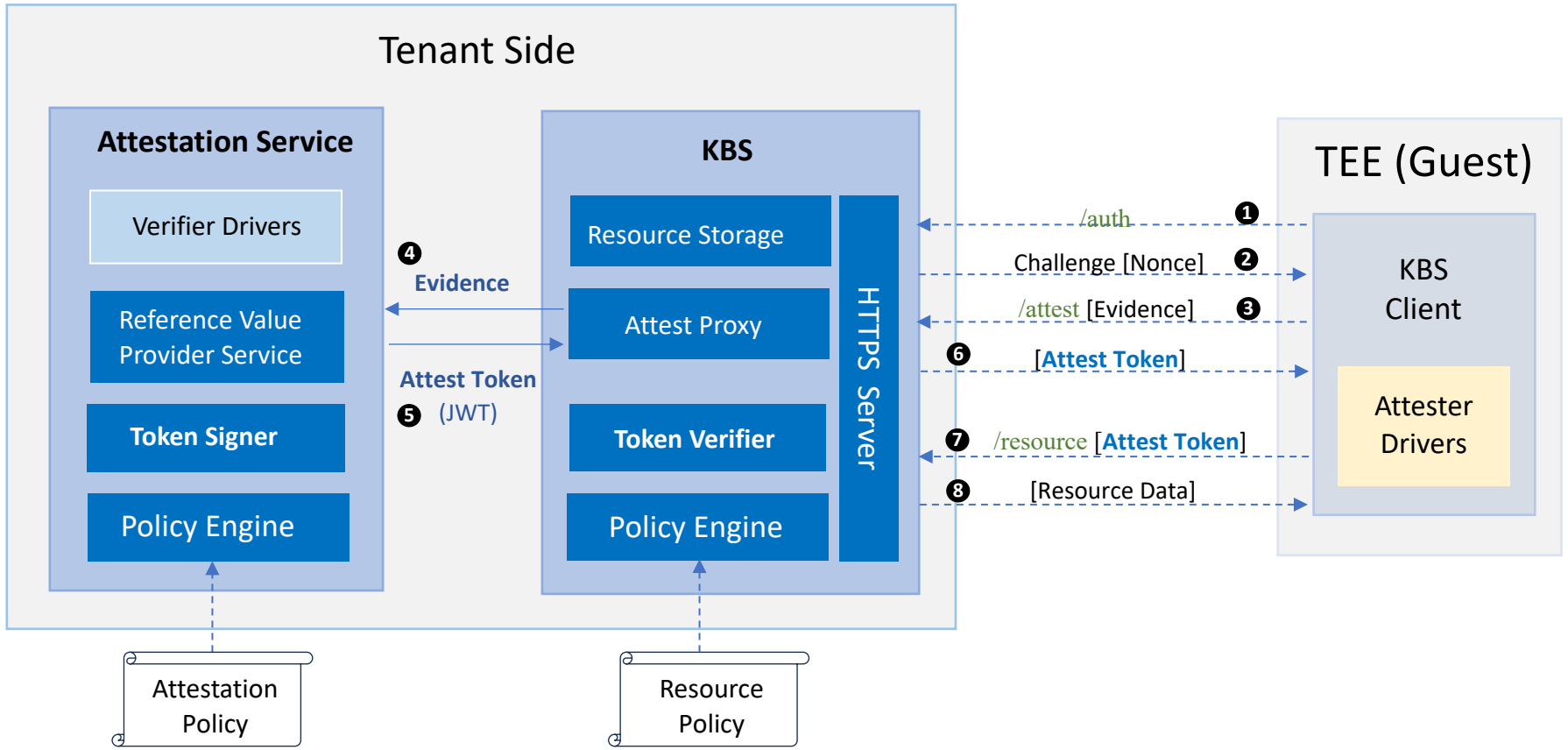
Projects:

- **Attestation Agent (AA)**: The **client side** of the Attestation system, running within TEE, integrates Attester Drivers for collecting TEE evidence, responsible for providing two API:
  1. obtaining TEE evidence
  2. obtaining Attestation Results Token from the remote Attestation Service
- **Key Broker Service (KBS)**: The **front-end server** of the Attestation system, which provides RESTful HTTPS APIs.
- **Attestation Service (AS)**: The **backend service** of the Attestation system, which is responsible for verifying TEE evidence and issuing Attention Results Token after verification is successful.

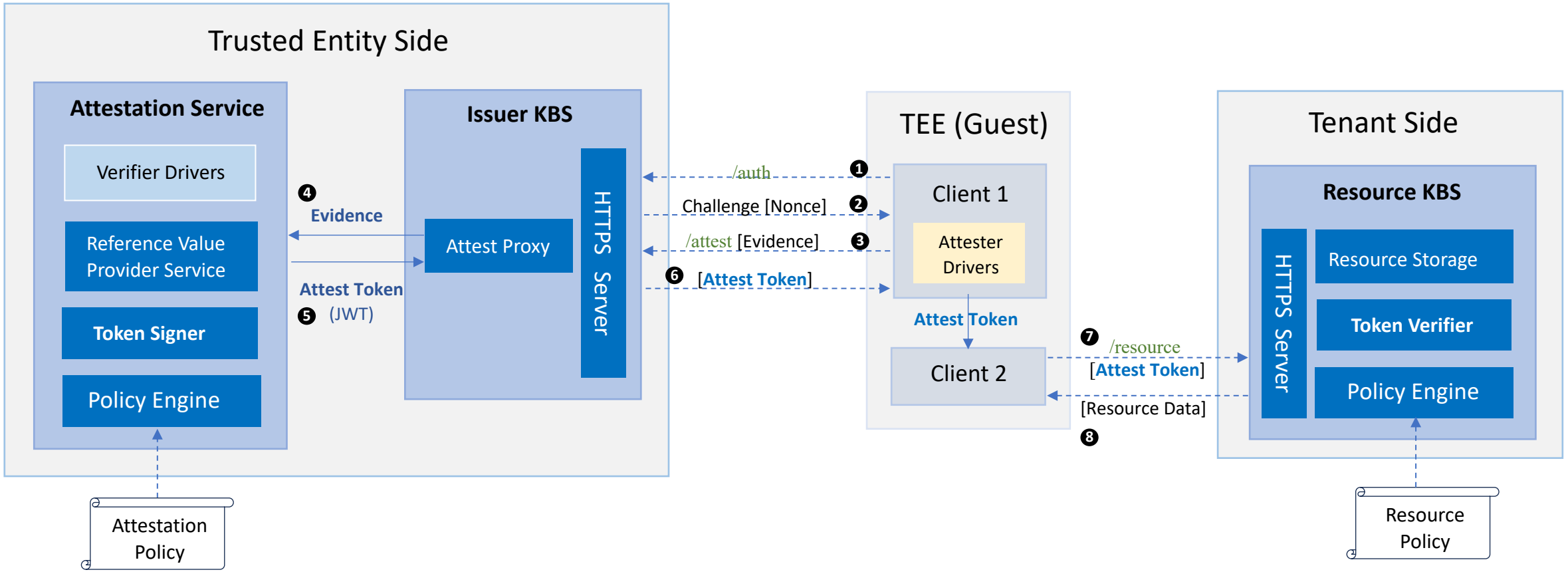
CoCo-AS Implementation



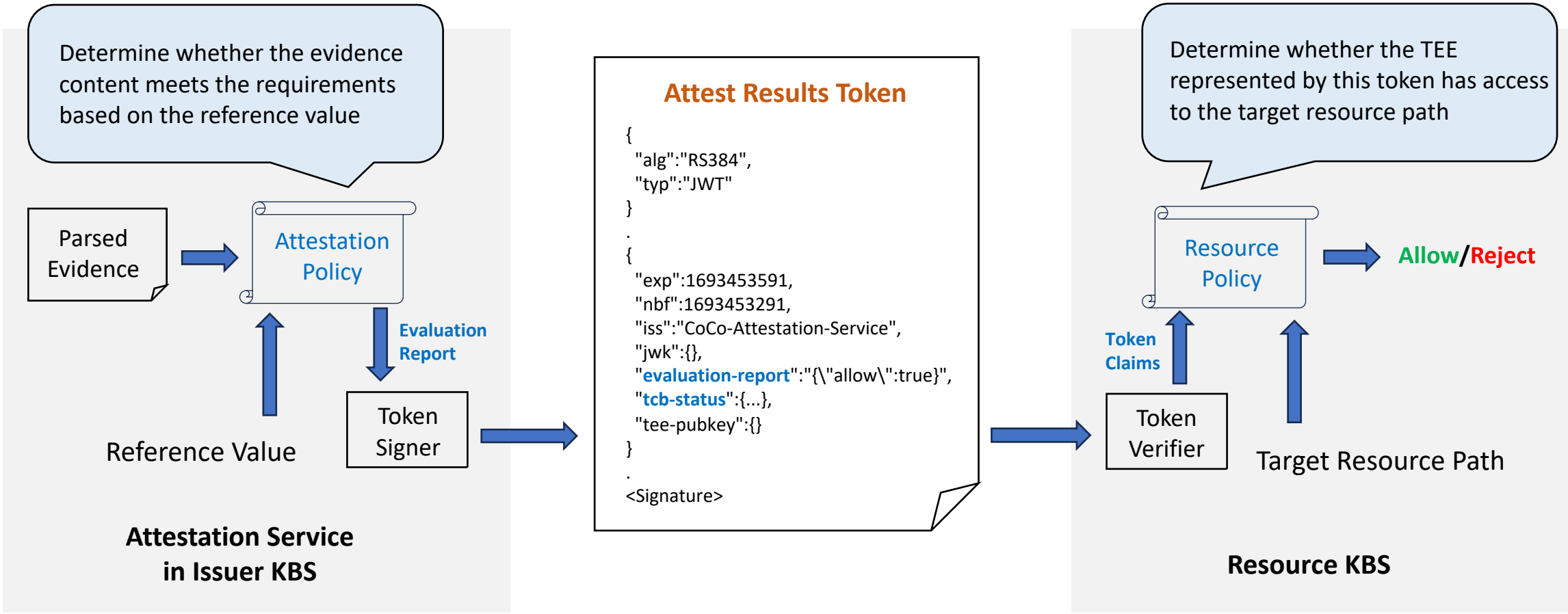
CoCo-AS Architecture - Mode1



CoCo-AS Architecture - Mode2

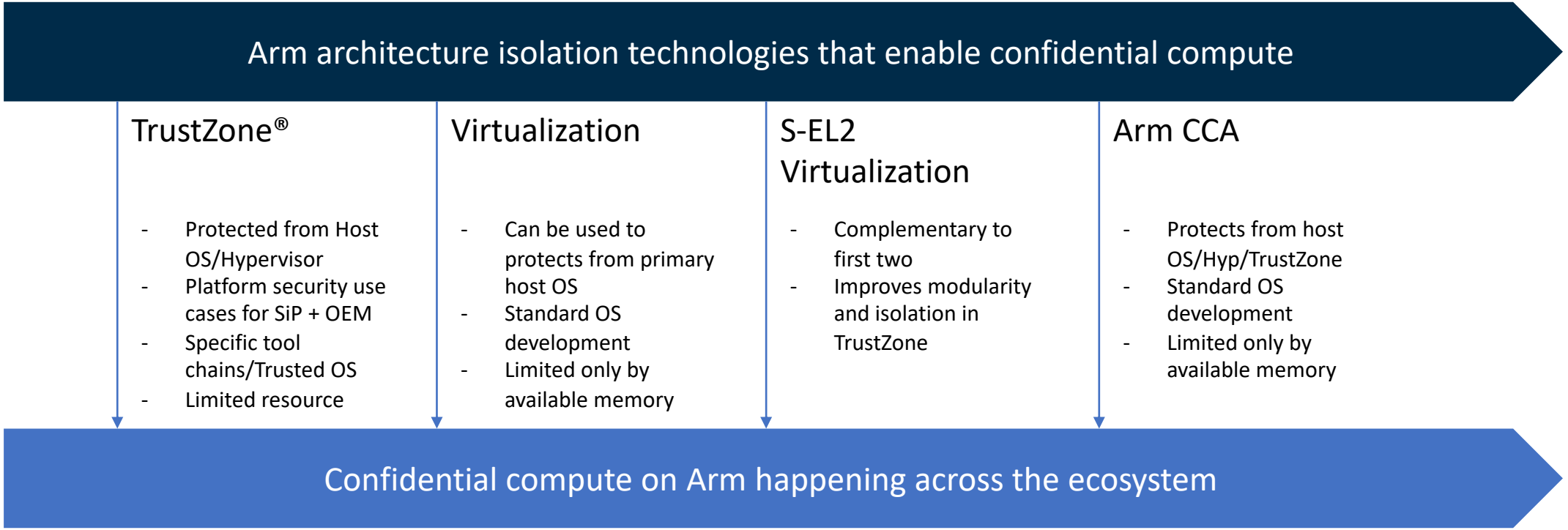


Policy & Token



Confidential Compute Architecture

Confidential Computing is part of a series of hardware and software architecture innovations that enhances Arm support for confidential computing, to shield portions of code and data from access or modification by privileged software and hardware agents

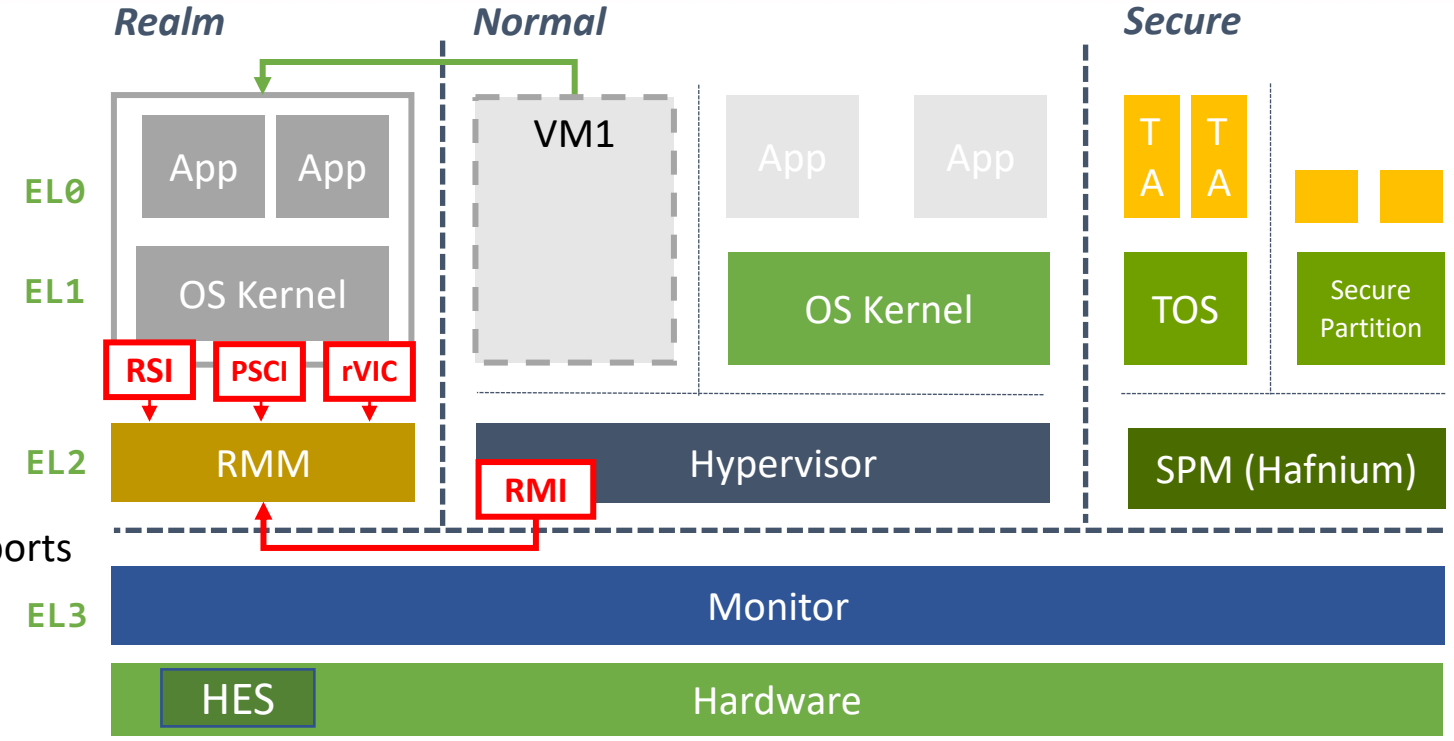


Confidential Compute Architecture - Overview

Realm Management Monitor (RMM)

- Realm Management Interface (RMI)
  - SMCCC interface called by Host
  - Create/destroy Realms
  - Manage Realm memory, manipulating stage 2 translation tables
  - Context switch between Realm VCPUs
- Realm Services Interface (RSI)
  - Primary use case is obtaining attestation reports
- EL3 Monitor
  - TF-A Monitor supporting FEAT_RME extension
  - Management of Granule Protection Table (GPT)

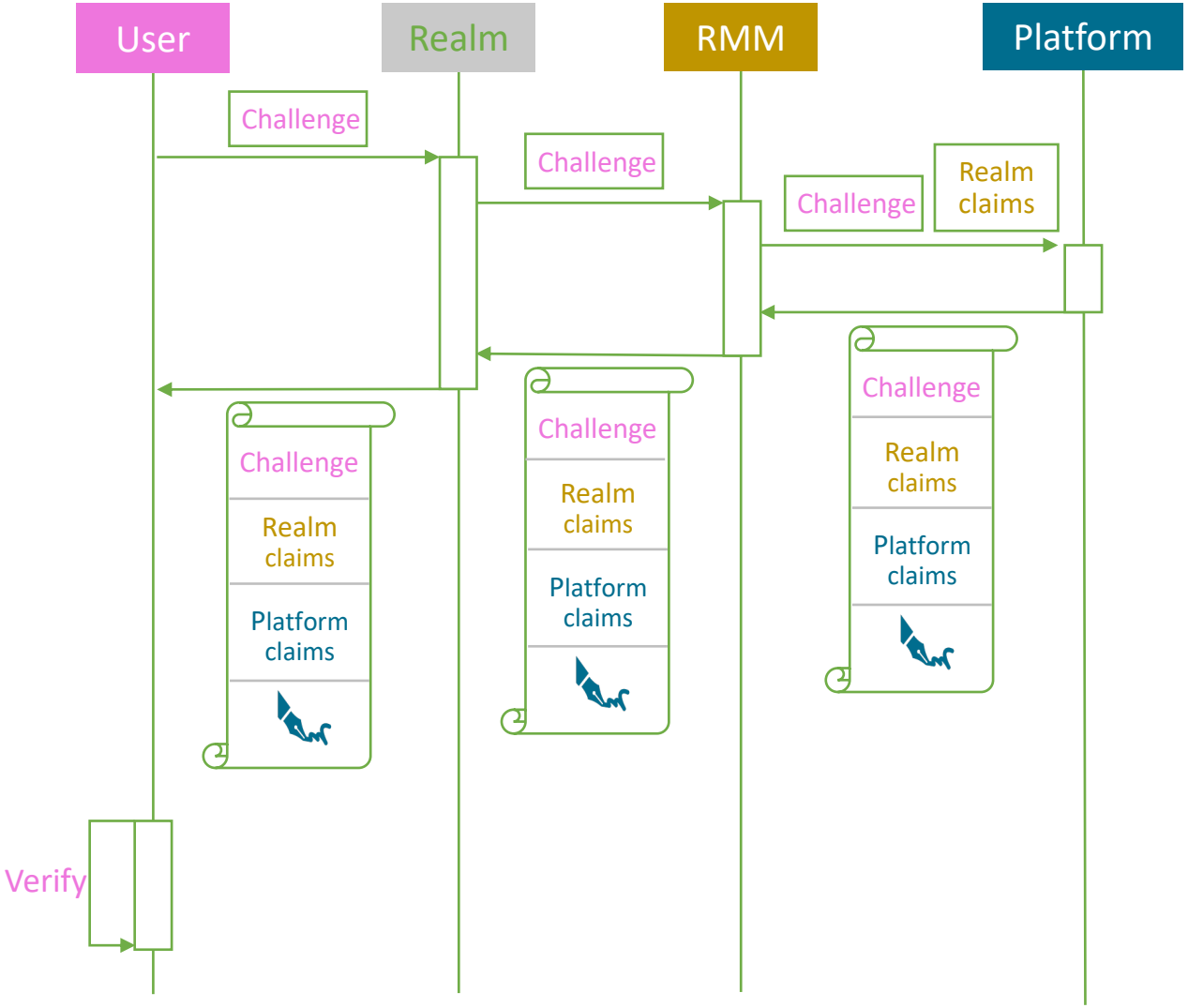
Controls assignment of memory to a Physical Address Space (PAS)



RSS (Runtime Security Subsystem)

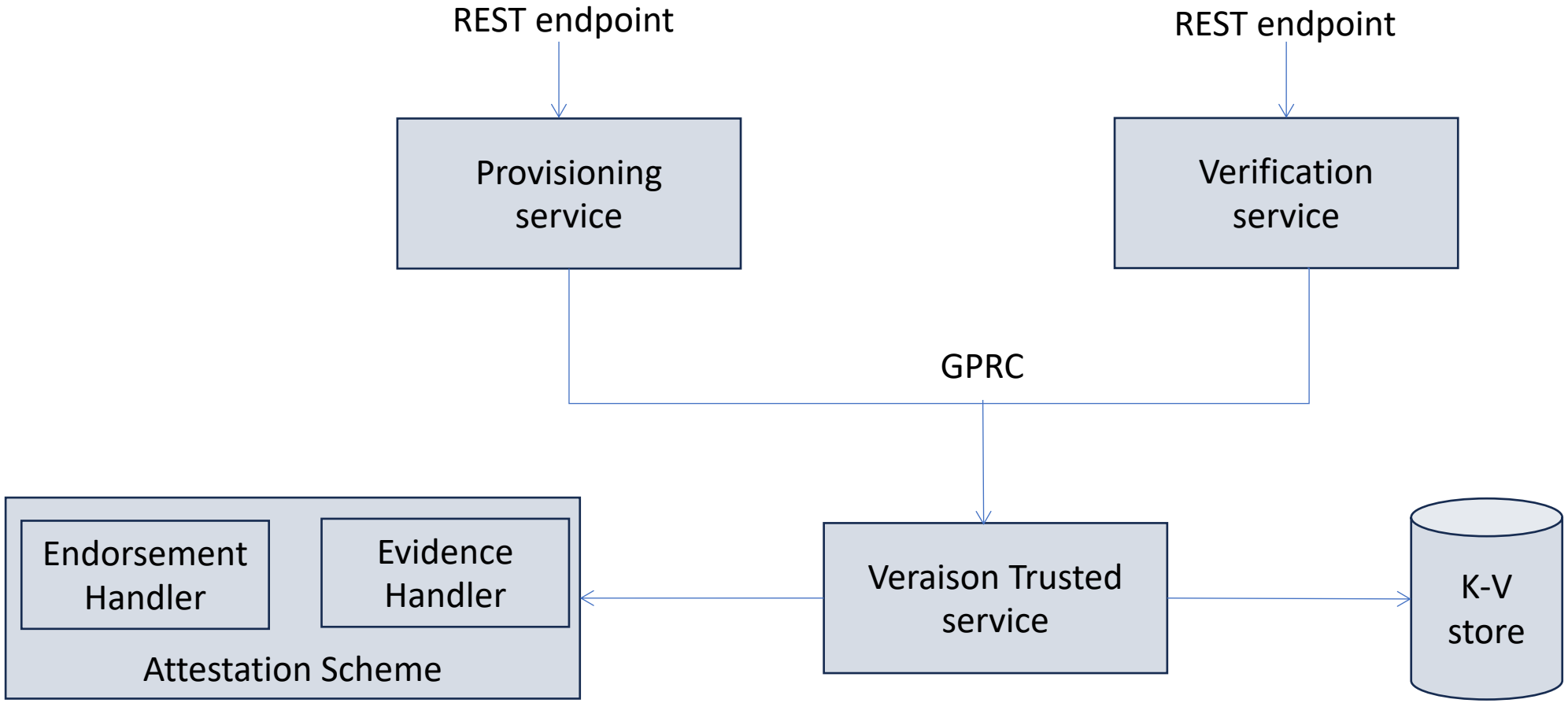
- Reference implementation of HES
- It serves as the Root of Trust for CCA
- M core

Confidential Compute Architecture – Attestation flow

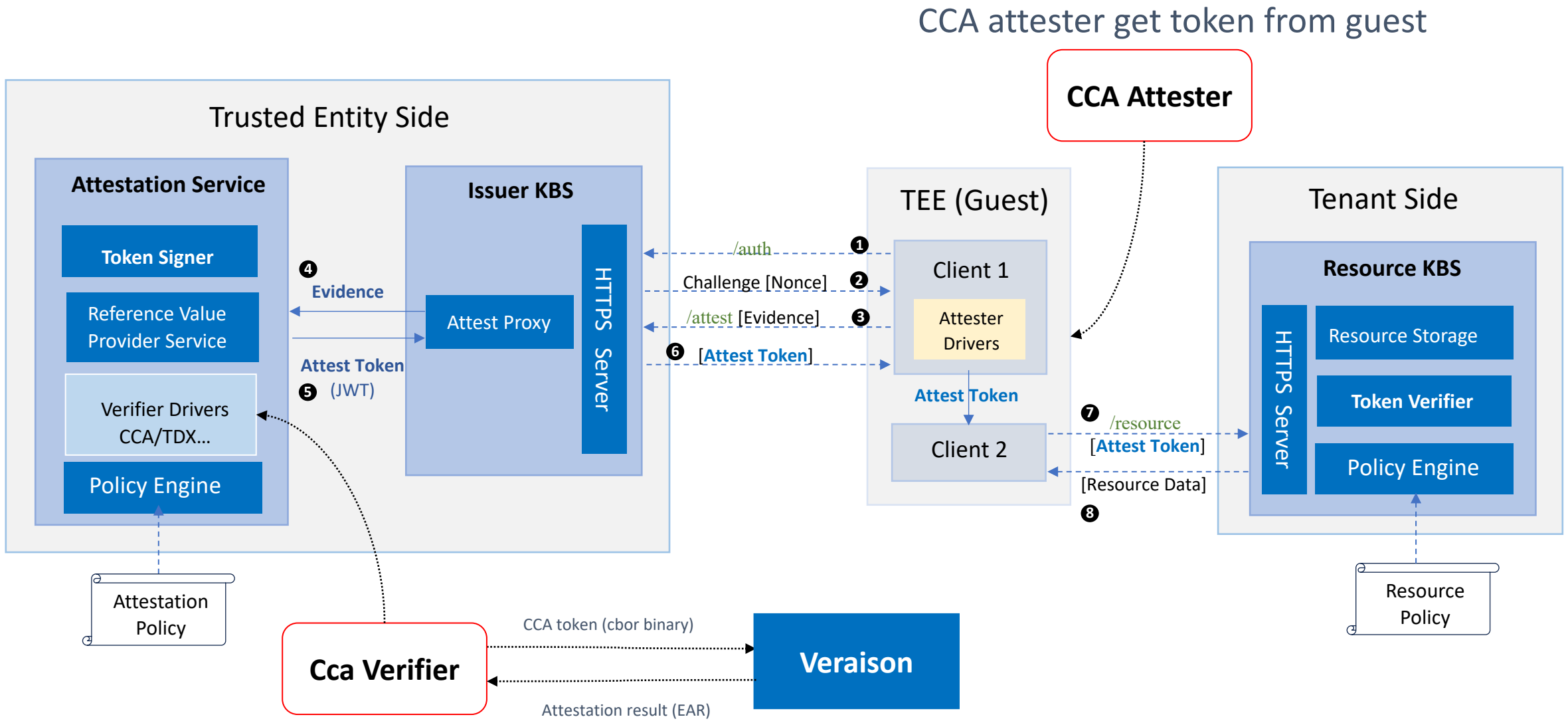


- Attestation report request includes a challenge, negotiated between the user and the Realm
- RMM generates Realm claims, including measurements taken during Realm creation
- Platform provides additional claims, including
  - Hardware identity
  - Firmware measurements and metadata
  - Whether external debug of the system can be enabled
- Report is encoded in a standard format
- Verification of the report involves
  - Checking cryptographic integrity
  - Comparing claims against known values
  - Comparing challenge against original request
- User may delegate this to a verification service
  - Arm is contributing to a project which provides components for constructing such a service
  - https://github.com/veraison

Veraison Architecture

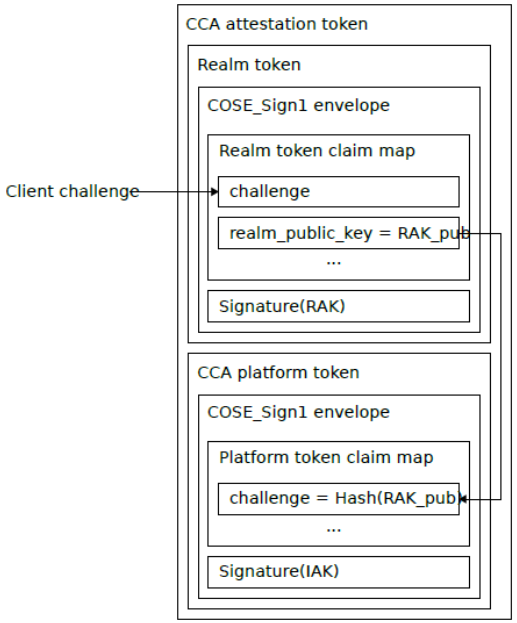


CoCo with CCA - Overview



CoCo with CCA – token layout

- Realm Management Monitor specification




```
{
  "cca-platform-token": {
    "cca-platform-profile": "http://arm.com/CCA-SSD/1.0.0",
    "cca-platform-implementation-id": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=",
    "cca-platform-instance-id": "AQICAgICAgICAgICAgICAgICAgICAgICAgICAgIC",
    "cca-platform-config": "AQID",
    "cca-platform-lifecycle": 12288,
    "cca-platform-sw-components": [
      {
        "measurement-description": "TF-M_SHA256MemPreXIP",
        "measurement-type": "BL",
        "measurement-value": "BwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "signer-id": "BwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "version": "3.4.2"
      },
      {
        "measurement-type": "M1",
        "measurement-value": "CwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "signer-id": "BwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "version": "1.2.0"
      },
      {
        "measurement-type": "M2",
        "measurement-value": "DwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "signer-id": "BwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "version": "1.2.3"
      },
      {
        "measurement-type": "M3",
        "measurement-value": "EwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "signer-id": "BwYFBAMCAQAPDg0MCwoJCBcWFRQTEhEQHx4dHBsaGRg=",
        "version": "1.0.0"
      }
    ],
    "cca-platform-service-indicator": "https://veraison.example/v1/challenge-response",
    "cca-platform-hash-algo-id": "sha-256"
  },
  "cca-realm-delegated-token": {
    "cca-realm-personalization-value": "QURBREFEQURBREFEQURBREFEQURBREFEQURBREFEQURBREFEQURBREFEQURBREFEQURBREFEQURBRA==",
    "cca-realm-initial-measurement": "Q0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQw==",
    "cca-realm-extensible-measurements": [
      "Q0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQw==",
      "Q0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQw==",
      "Q0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQw==",
      "Q0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQw=="
    ],
    "cca-realm-hash-algo-id": "sha-256",
    "cca-realm-public-key": "BIL70TKptcOWh5+7FTQNkFCXjlXHnVJ5oroOlYVPN+IM0vZPO3K1cLvXc+7iznaEJe31Re2+if+v4OlrvUbicPIHlsRIuY2vRqdk0nRC5ubthPjOyBfm7ManHTo959Z+zQ==",
    "cca-realm-public-key-hash-algo-id": "sha-512"
  }
}
```

CoCo with CCA – token generation

- RFC (CCA support for Linux) can be found from here: https://www.spinics.net/lists/kvm/msg302232.html
- (CCA driver) Initialize CCA module and create a character device with open/write/ ioctl interfaces defined.
- ioctl function *cca_attestation_request* will fetch the CCA token via realm service interface (RSI).
- The token is cbor formatted binary, realm token and platform token are signed with Realm Attestation Key(RAK) and Initial Attestation Key(IAK) respectively.
- Send the CCA token to attestation service for verification.
- Token generation is now available in a CCA simulation environment, e.g., FVP.

CoCo with CCA – Verification

- Build an endpoint and send the CCA token to Veraison for verification
  - Parse by the token by fxamacker/cbor and ccatoken.
  - Integrity check
    - Verify the signature of the platform token by the IAK (initial attestation key)
    - Verify the signature of the realm token by the RAK (realm attestation key)
    - Check the key binding between the IAK and RAK – hash(rak) == platform token::challenge
  - Evidence appraisal
    - Compare realm evidence and platform evidence with the reference value defined in the k-v store.
  - Policy enforcement
    - Similar with CoCo (OPA Policy, Rego file)
- Parse the CCA token by cbor diag, for debug only!
  - Limitation: Cannot parse the token to a json file with rust lib
  - Compare CoCo's challenge with the challenge sealed in the realm token - hash(nonce||pubkey) == realm token::challenge

Booth No. S10

The Future is Built on arm

arm TECH SYMPOSIA

Arm 年度技术大会

11月27日 深圳 | 11月29日 北京 | 12月1日 上海

关注 “Arm 社区” 公众号

与 1500 万志同道合者

在 Arm 平台上构建未来