



**CONFIDENTIAL  
COMPUTING  
SUMMIT 2023**

**vmware**

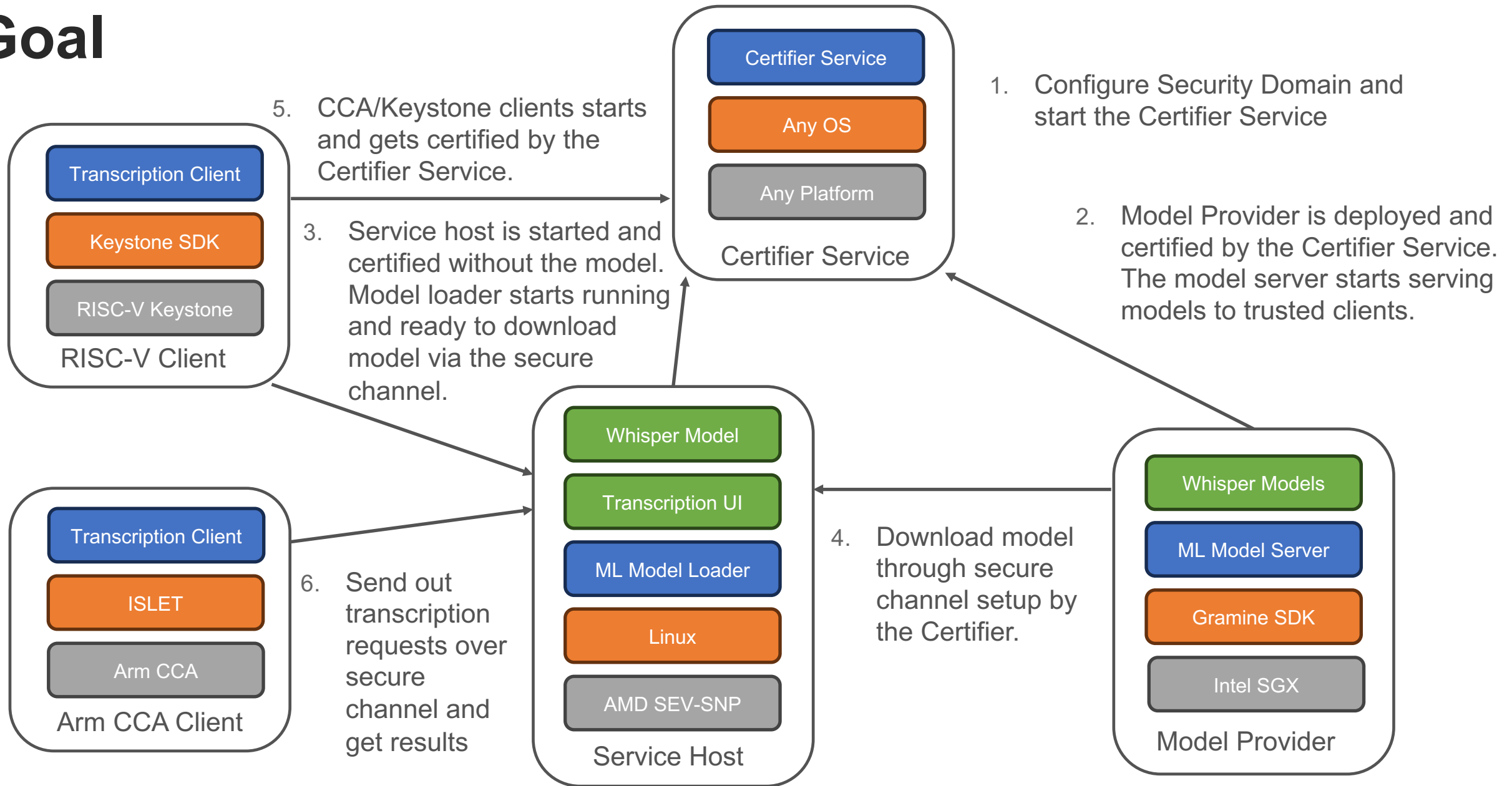
# **An Open-Source Certifier Framework for Confidential Computing**

**John Manferdelli, CC Project Director**

**Ye Li, Sr. Engineer**

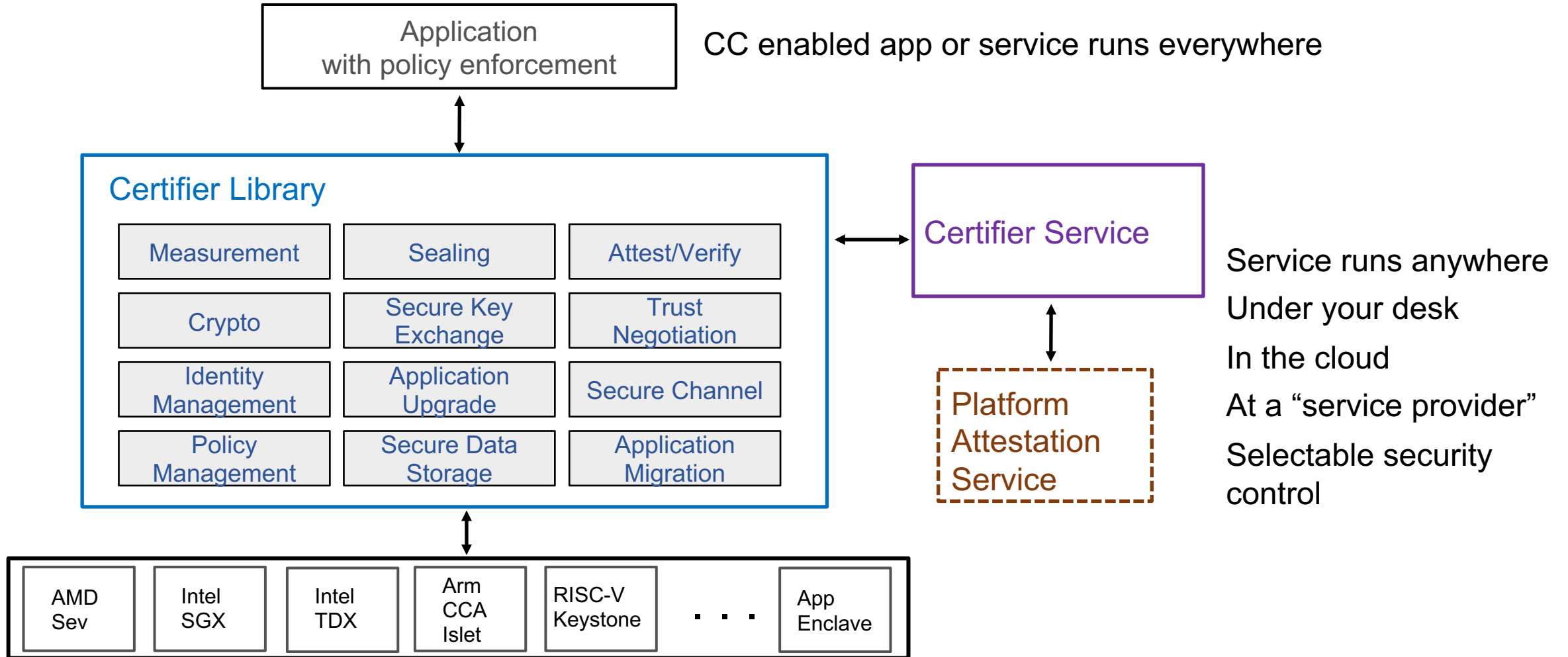
**June 29, 2023**

# Goal



# The Certifier Framework for Confidential Computing

Taking the devil out of the details in CC software development



# Certifier Concepts and API

## Key Concepts:

### Security Domain

Identified by a public key associated with all application code within a trusted environment.

### Certification

Refers to the verification of all properties in the security domain (including program identity, involving attestation) resulting in an x509 certificate for trust.

### Trust and Policy

Should not be hard-coded in an application which should be able to operate in compliance in different security domains. Don't complicate program development or deployment.

## C++ Classes:

<code>cc_trust_data</code>	Basic interface to establish keys, policy and manage certification with the certifier service.
<code>secure_authenticated_channel</code>	Establishes secure channel with an “authenticated program in security domain
<code>policy_store</code>	Stores policy, keys, communications endpoints securely. Additional helper function APIs for complicated applications.

Additional helper function APIs for complicated applications.

# Most applications will use exactly this (and nothing else)

```
string public_key_alg("rsa-2048"); string symmetric_key_alg("aes-256-gcm");  
cc_trust_data app_trust_data("sev-enclave", "authentication", "store");
```

```
app_trust_data.initialize_sev_enclave_data(...); // init enclave  
app_trust_data.init_policy_key(initialized_cert_size, initialized_cert)  
app_trust_data.cold_init(public_key_alg, symmetric_key_alg, ...);  
app_trust_data.certify_me(); // Get admission certificate
```

```
secure_authenticated_channel channel("client");  
channel.init_client_ssl(host, port, policy-key, auth_key, admissions-cert);
```

```
// Your application here
```



# Certifier: “Simple Example” App

Running App as server

Client peer id is Measured

8522d8e996e0089b26cb5dde06341c728e3017fa78974e3dce364bef56bdd307

SSL server read: Hi from your secret client

Running App as client

Server peer id : Measured

8522d8e996e0089b26cb5dde06341c728e3017fa78974e3dce364bef56bdd307

SSL client read: Hi from your secret server

You may want to add:

- API serialization
- ACLs for differentiated access
- Standard “distributed programming” primitives

## Works on all CC platforms:

- Simulated enclave
- AMD Sev-SNP
- Intel SGX (Gramine and OE)
- Arm CCA (Samsung Islet)
- RISC-V Keystone
- Intel TDX (when it’s ready)
- Application enclave service

# Certifier API: Observations

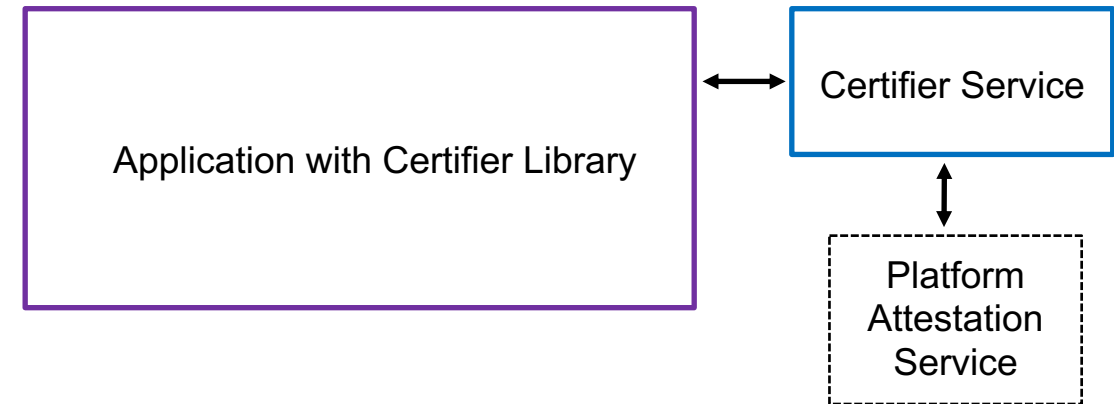
1. **Simple** for most applications (and **adequate** for complex ones)
2. Abstracts underlying **isolate**, **measure**, **seal/unseal**, and **attest** primitives
3. Provides a **secure store** for save/recovery operations (in one statement!).
4. Includes mechanism to establish a **secure channel** within security domain
  - Encrypted, integrity protected, bi-directional, with authenticated trusted enclave named by their measurements
5. Almost as simple as **“Hello world”**

# Certifier Service

“Centralized” management for a security domain

- Policy-driven (rooted in key associated with application)
- Admits new components without changing old ones
- Application upgrade without changing other applications
- Facilitates data migration and sharing in a domain
- Enforce security domain wide policy
- Machine capabilities
- Revocation

*Scalable, resilient deployment supporting all environments*



## Policy example

1. Key[rsa, **policyKey**, a5fc2b7e629fbbfb04b056a993a473af3540bbfe] is-trusted
2. Key[rsa, **policyKey**, ... ] says **Measurement**[a051a41593ced366462caea392830628742943c3e81892ac17b70dab6fff0e10] is-trusted
3. Key[rsa, **policyKey**, ...] says Key[rsa, **platformKey**, ...] is-trusted-for-attestation
4. Key[rsa, **platformKey**, ...] says Key[rsa, **attestKey**, ...] is-trusted-for-attestation



# Proof from the Certifier Service

1. `Key[rsa, policyKey, ...]` is-trusted and `Key[rsa, policyKey, ...]` says `Measurement[a051a41593ced366462caea392830628742943c3e81892ac17b70dab6fff0e10]` is-trusted, imply via rule 3, `Measurement[...]` is-trusted
2. `Key[rsa, policyKey, ...]` is-trusted and `Key[rsa, policyKey, ...]` says `Key[rsa, platformKey, ...]` is-trusted-for-attestation, imply via rule 5, `Key[rsa, platformKey, ...]` is-trusted-for-attestation
3. `Key[rsa, platformKey, ...]` is-trusted-for-attestation and `Key[rsa, platformKey, ...]` says `Key[rsa, attestKey, ...]` is-trusted-for-attestation, imply via rule 5, `Key[rsa, attestKey, ...]` is-trusted-for-attestation
4. `Key[rsa, attestKey, ...]` is-trusted-for-attestation and `Key[rsa, attestKey, ...]` says `Key[rsa, program-auth-key, ...]` speaks-for `Measurement[...]`, imply via rule 6, `Key[rsa, program-auth-key, ...]` speaks-for `Measurement[...]`
5. `Measurement[...]` is-trusted and `Key[rsa, program-auth-key, ...]` speaks-for `Measurement[...]`, imply via rule 1, `Key[rsa, program-auth-key, ...]` is-trusted-for-authentication



**Proved:** `Key[rsa, program-auth-key, ...]` is-trusted-for-authentication

# Platform Policy

Used to verify platform characteristics

- Key[rsa, policyKey, a5fc2b7e629fbbfb04b056a993a473af3540bbfe] says Key[rsa, ARKKey, aeb214025256a56863fe9aa9c9f1cca153af4416] is-trusted-for-attestation
- Key[rsa, policyKey, a5fc2b7e629fbbfb04b056a993a473af3540bbfe] says platform[amd-sev-snp, debug: no, migrate: no, api-major:  $\geq 0$ , api-minor:  $\geq 0$ , smt: no, tcb-version: = 3458764513820573973] has-trusted-platform-property

Supplants the need to use external attestation services and preserves privacy and control.

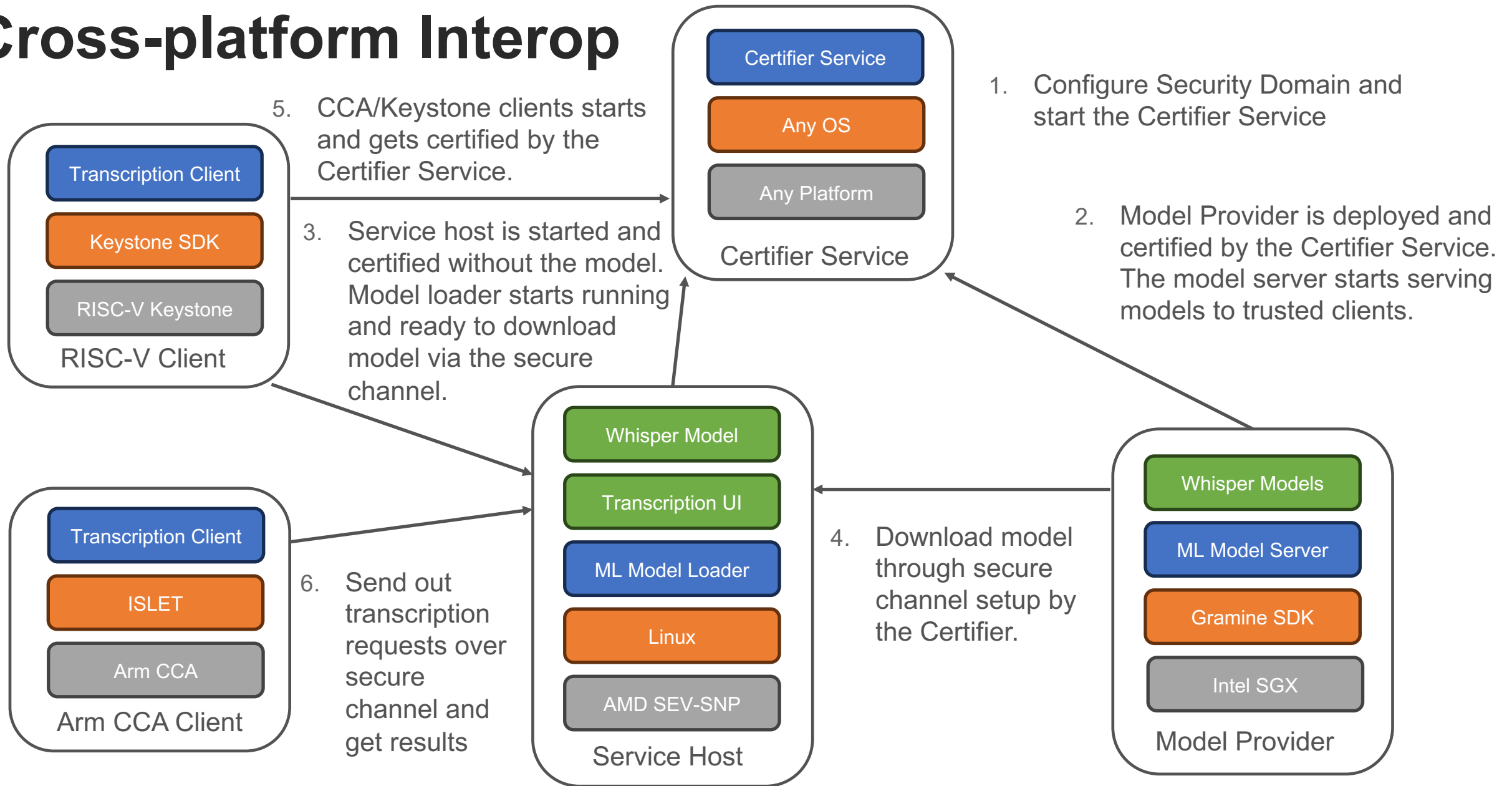
# Certifier Service: Observations

1. Service provides a **policy language**, **evidence formats**, and **policy evaluation** to determine when a Confidential Computing application should be trusted.

Evidence submitted and evaluated includes platform attestation reports. Other formats converted to “canonical form.”

2. Utilities to **generate keys** and **write policy**.
3. Checks **program** and **platform policy**
4. Issues “**Admissions Certificate for Security Domain**”

# Cross-platform Interop



# More to come...

- Security review
- GPU support
- Python bindings
- Rust Client

# Join the Party...



## Open-Source project

[github.com/vmware-research/certifier-framework-for-confidential-computing](https://github.com/vmware-research/certifier-framework-for-confidential-computing)

Apache license

Contributions and new contributors are welcome

Make Confidential Computing an open Universal Platform

A screenshot of the GitHub repository page for 'vmware-research/certifier-framework-for-confidential-computing'. The page shows the repository name, a 'Public' badge, and navigation links for Code, Issues (6), Pull requests (7), Actions, Projects, Security, and Insights. It also displays the current branch (main), 20 branches, and 1 tag. A list of recent commits is shown, including one by 'gapisback' (#111) and another by '#106' for integrating build-and-test. The 'About' section describes the Confidential Computing Certifier Framework.

Product ▾ Solutions ▾ Open Source ▾ Pricing

Search / Sign in Sign up

vmware-research / certifier-framework-for-confidential-computing Public

Notifications Fork 9 Star 15

<> Code Issues 6 Pull requests 7 Actions Projects Security Insights

main ▾ 20 branches 1 tag Go to file Code ▾

**gapisback (#111)** Fix few utility programs, to print error messages for invalid... ✓ b866412 yesterday 1,044 commits

**#106** Integrate build-and-test of Keystone app under run\_example... last week

**#93** Support simple app under sev to run in simulated SEV mode. 2 weeks ago

**About**

The Confidential Computing Certifier Framework consists of a client API called the certifier-API and server based policy evaluation called the certifier Service. It simplifies and unifies programming and



# Thank you!

John Manferdelli <jmanferdelli@vmware.com>

# An Open-Source Certifier Framework for Confidential Computing

**Abstract:** Confidential Computing is a foundational technology, but its adoption has been inhibited by the difficulty in implementing programs quickly even on a single platform. In addition, fragmentation in the TEE platform market has prevented software portability and reuse across TEE technologies. In this session, we will discuss the *Certifier Framework for Confidential Computing*, an open-source project offering a simple, general API and accompanying service for managing attestation in scaled CC programs. With a half dozen or so API calls, a developer can incorporate CC into their software without deep expertise in security and platform-specific TEE technologies. Furthermore, the framework also decouples trust policy from program code and supports managed deployment. We'll cover the programming model, trust model and support (including policy and key storage) that makes the Certifier Framework easy to use and broadly applicable.

# Summary: Removing Barriers with CC + Certifier

Verifiably secure operational properties (confidentiality, integrity, policy compliance) no matter where a program runs. Safe against malware and “insiders.”

## Before CC: developer/deployer must

1. Write applications correctly
2. Deploy the program safely (no changes)
3. Configure operating environment correctly
4. Ensure other programs can't interfere with safe program execution
5. Generate and deploy keys safely
6. Protect keys during use and storage
7. Ensure data is not visible to adversaries and can't be changed in transmission or storage
8. Ensure trust infrastructure is reliable
9. Audit to verify this all happened

### Consequence:

- App writer/deployer entirely reliant on provider for all security --- unverifiable

## With CC: developer/deployer must

1. Write the application correctly
  - For every backend
  - Manage migration
  - Support each providers deployment model
  - Implement all the crypto
  - Implement secure communications and storage
  - Make it scalable and upgradable
2. Implement the trust policy
  - Maintain trust policy
  - Different for every app/deployer
  - Make it scalable

### Consequence:

- You can have safe application but it's platform dependent and a lot of work

## With CC + Certifier: developer/deployer must

1. Write the application correctly using VMware APIs
2. Write the trust policy
3. Use Certifier Service to manage it!

### Consequence:

- You write the application once.
- Need only add a few dozen lines of code to enable CC protection.
- Trust policy is independent of application.
- Can move to another “backend” effortlessly