OPEN POSSIBILITIES.

Platform Integrity Attestation at Scale

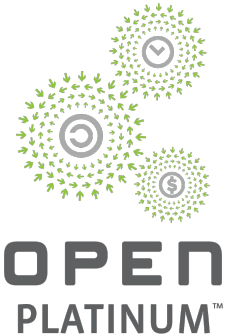
OCP GLOBAL SUMMIT
NOVEMBER 9-10, 2021

SECURITY

Platform Integrity Attestation at Scale

Jeff Andersen - Google, SWE

Jonathan Cooke - Google, SWE



OPEN POSSIBILITIES.

NIST 800-193: Platform Firmware Resiliency

SECURITY

- **Protection** - Ensuring that Platform Firmware code and critical data remain in a state of integrity and are protected from corruption
- **Detection** - Detecting when Platform Firmware code and critical data have been corrupted or otherwise changed from an authorized state
- **Recovery** - Restoring Platform Firmware code and critical data to a state of integrity in the event that it is detected to have been corrupted

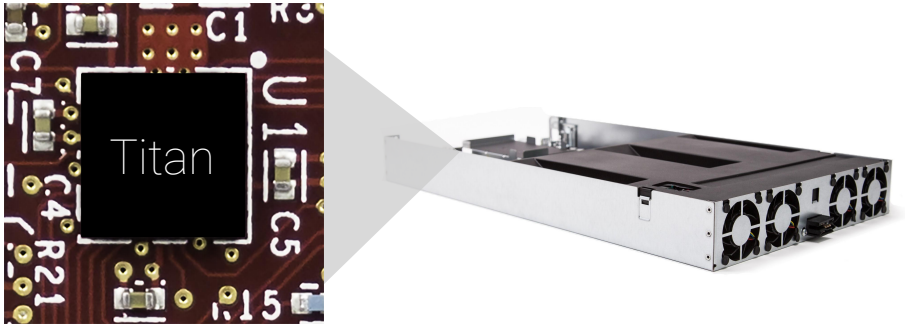
Integrating *detection* into machine fleets

SECURITY

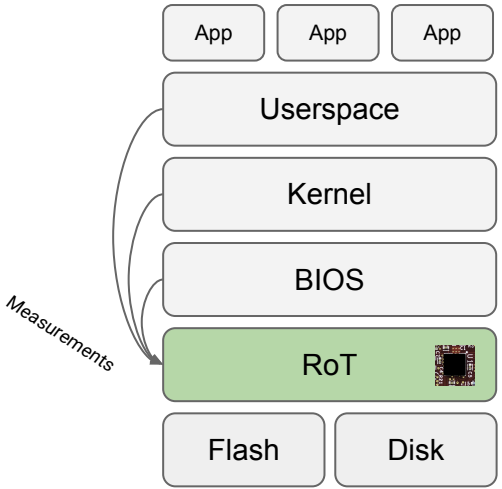
- **Policymaking** - Describing the platform's intended code and data configuration
- **Routing** - Exposing attestation signals to a verifier
- **Reaction** - Ensuring that appropriate action be taken when policies are violated

In the beginning, there was Titan

SECURITY



Complexity of yesterday

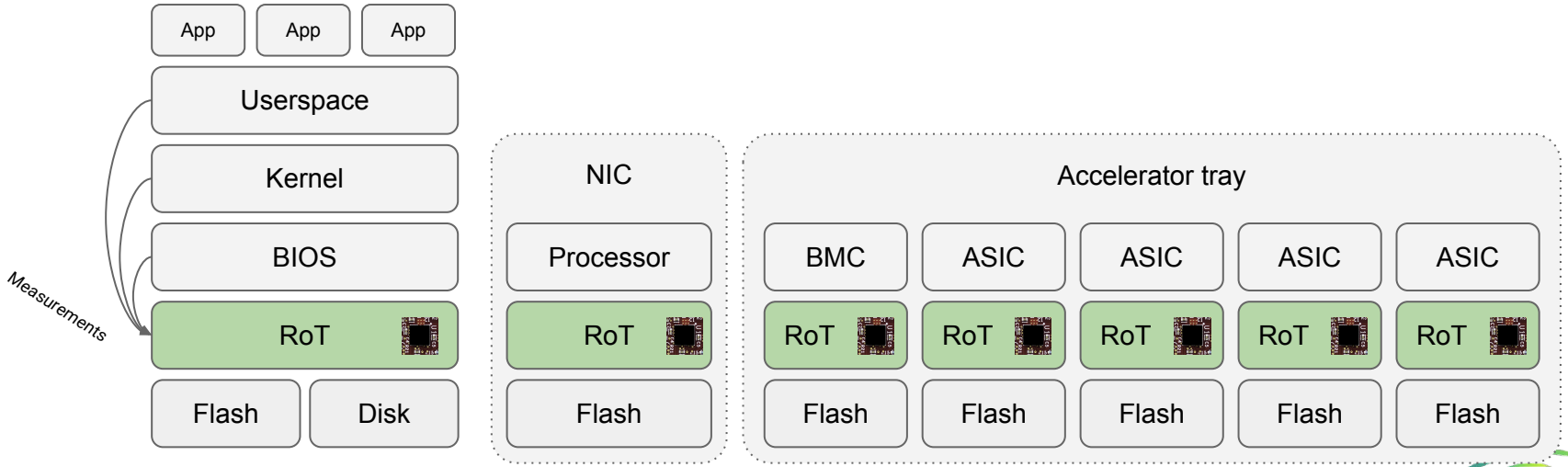


SECURITY

- **Root of trust (RoT)** - Secure element with a strong cryptographic identity
  - **Measurer** - Actively takes measurements of flash chip contents
  - **Attester** - Wields cryptographic identity to report trustworthy measurements
- Two strong assumptions:
  - **1:1 relationship between machines and RoTs**
  - **Everything of interest is measured into the single RoT**

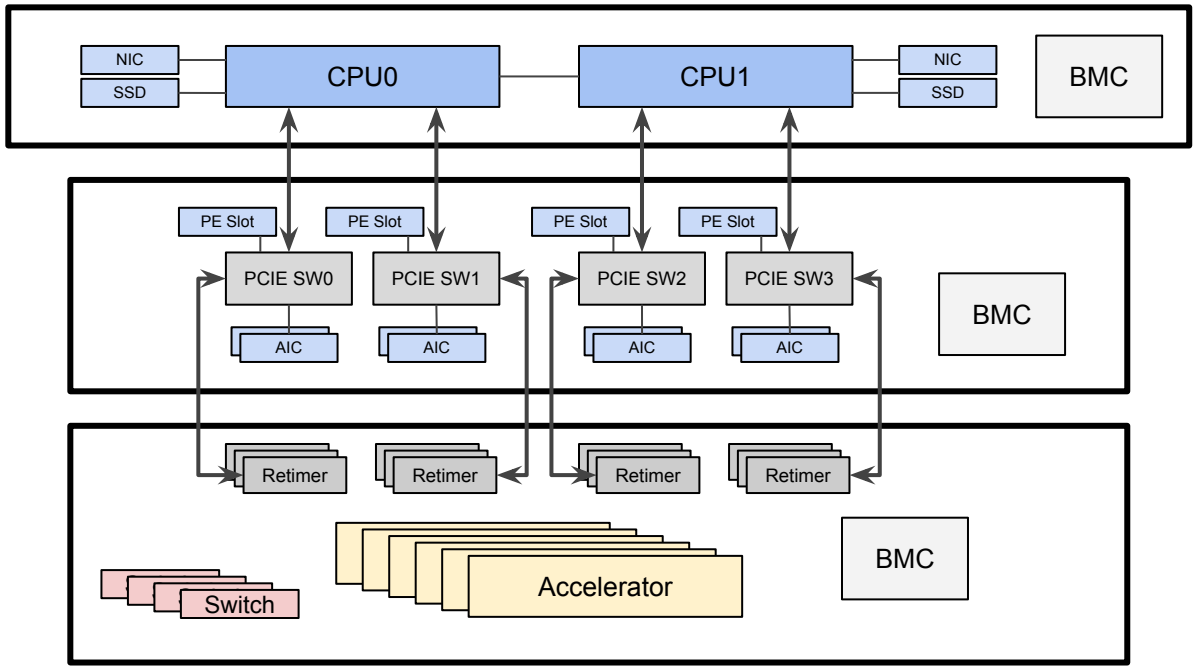
Complexity of today

SECURITY



OPEN POSSIBILITIES.

Complexity of tomorrow



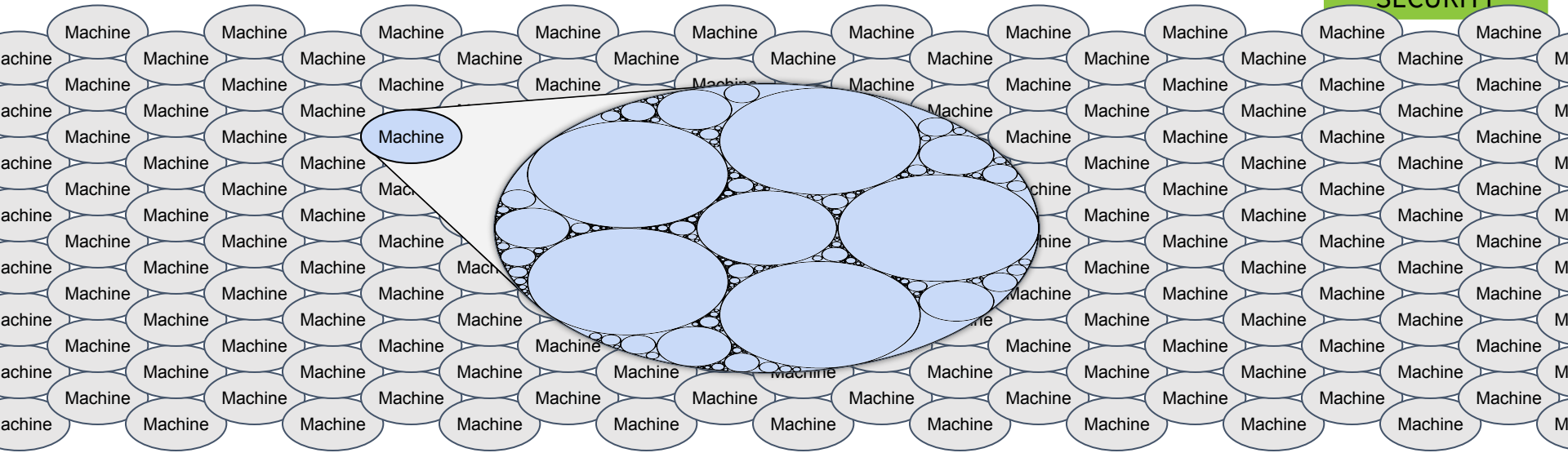
SECURITY

~20× RoTs

3x BMCs & power domains

AIC = Arbitrary Add-in Card

Attestation "at scale"



OPEN POSSIBILITIES.

OCP GLOBAL SUMMIT
NOVEMBER 9-10, 2021

So what are we sharing today?

SECURITY

- **Design constraints** - If they sound familiar, this scheme might be right for you
- **Our attestation solution** - Scalable middleware for attesting complex machines
- **Some standards gaps** - What we invented ourselves to get it to tick

Google fleet management philosophy

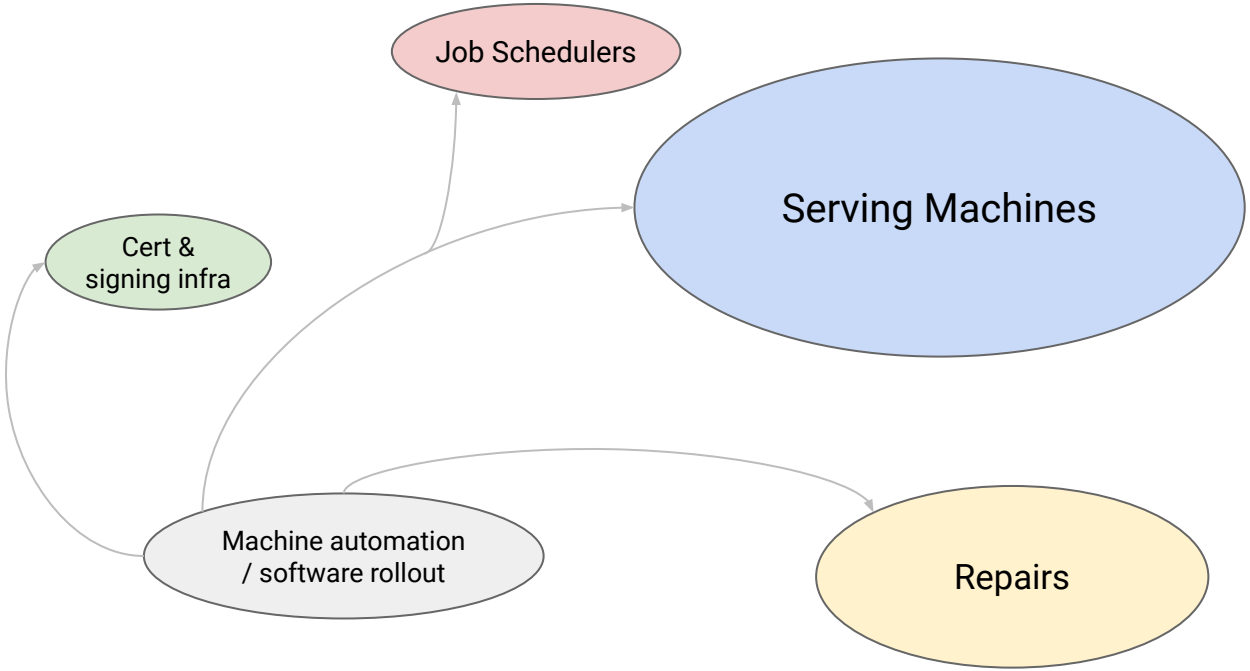
SECURITY

- **Production is in a constant state of flux**
  - Software always rolling out, machines breaking, getting fixed...
  - Nothing is perfect: sometimes updates don't take, machines are incorrectly repaired, etc.

- **Risk of outages must be minimized**
  - E.g., machines must not brick themselves en masse
  - Where possible, prefer complexity in the control plane rather than on-machine

- **"Machines are cattle, not pets"**
  - Individual machines are allowed to die, but we like to know what killed them
  - Automate, automate, automate

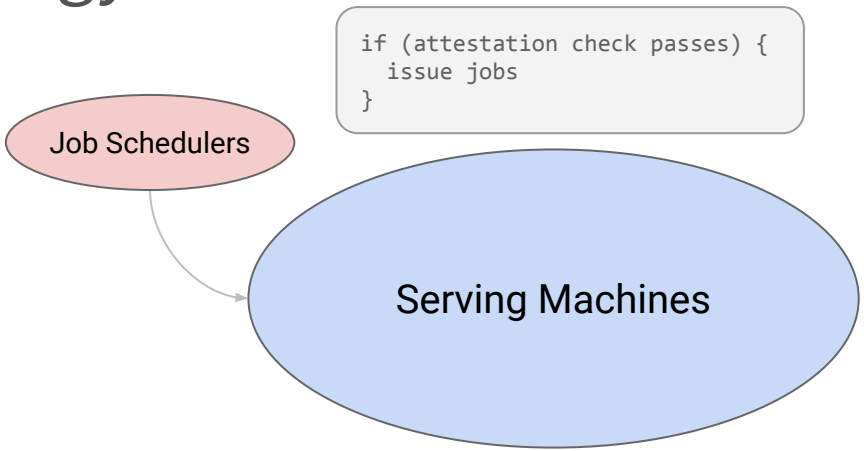
Google's production fleet



SECURITY

(Not drawn to scale)

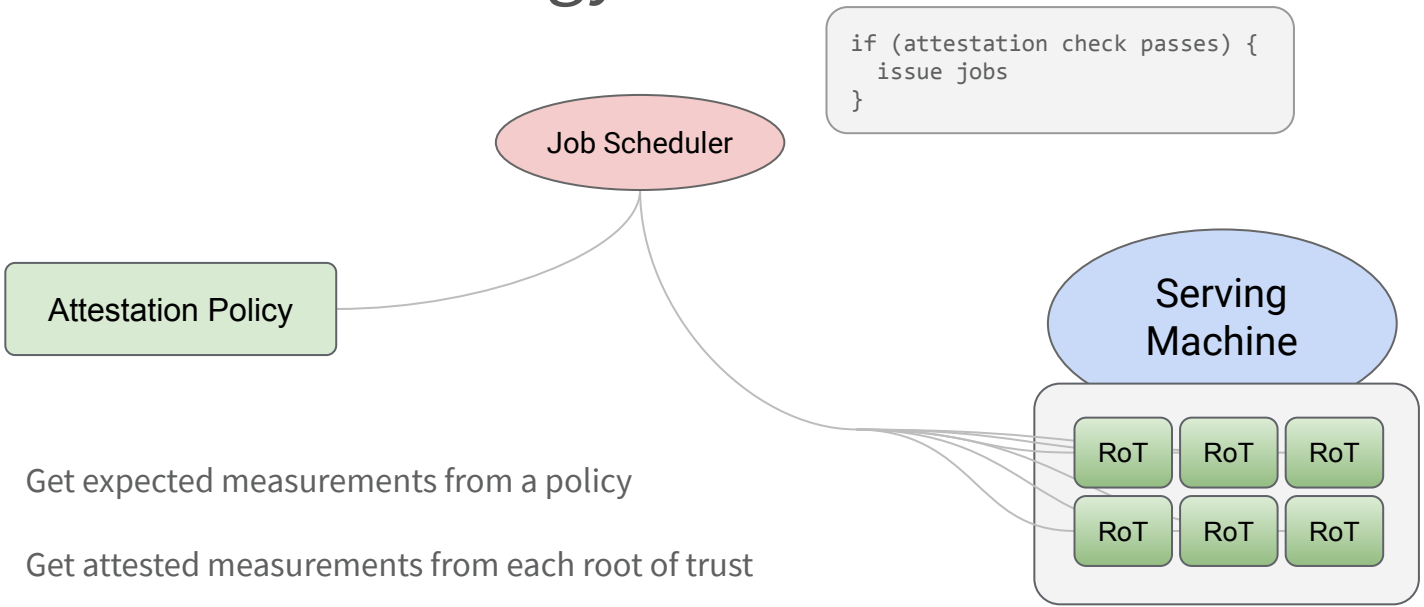
Attestation strategy



SECURITY

- **Desired property**: Jobs only run on machines booting intended firmware
- **Strategy**: Job schedulers issue attestation challenges to machines
  - Only issue jobs and data on successful attestation

Attestation strategy



SECURITY

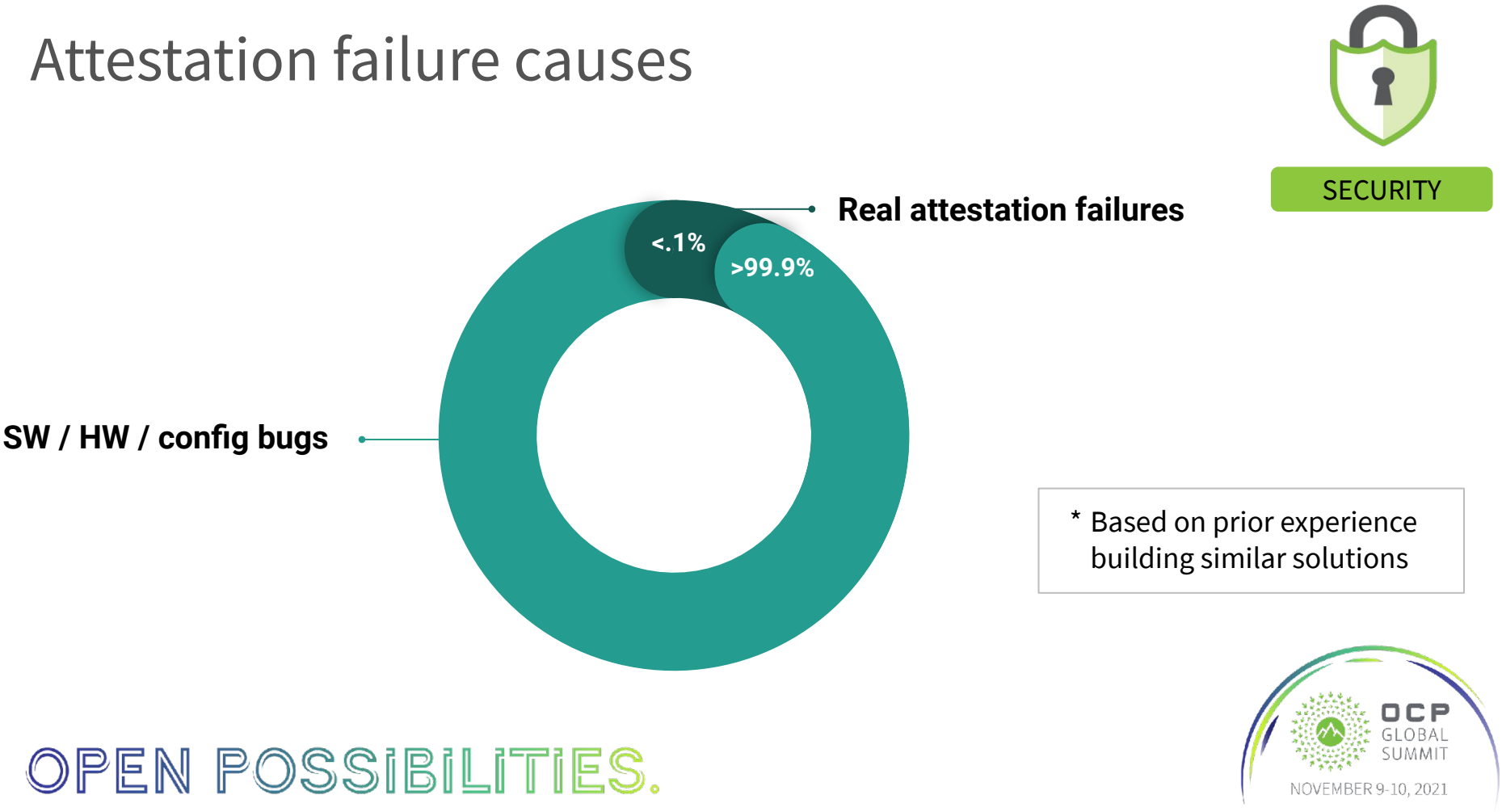
1. Get expected measurements from a policy
2. Get attested measurements from each root of trust
3. Compare attestations to expectations

Why not enforce the policy on-machine?

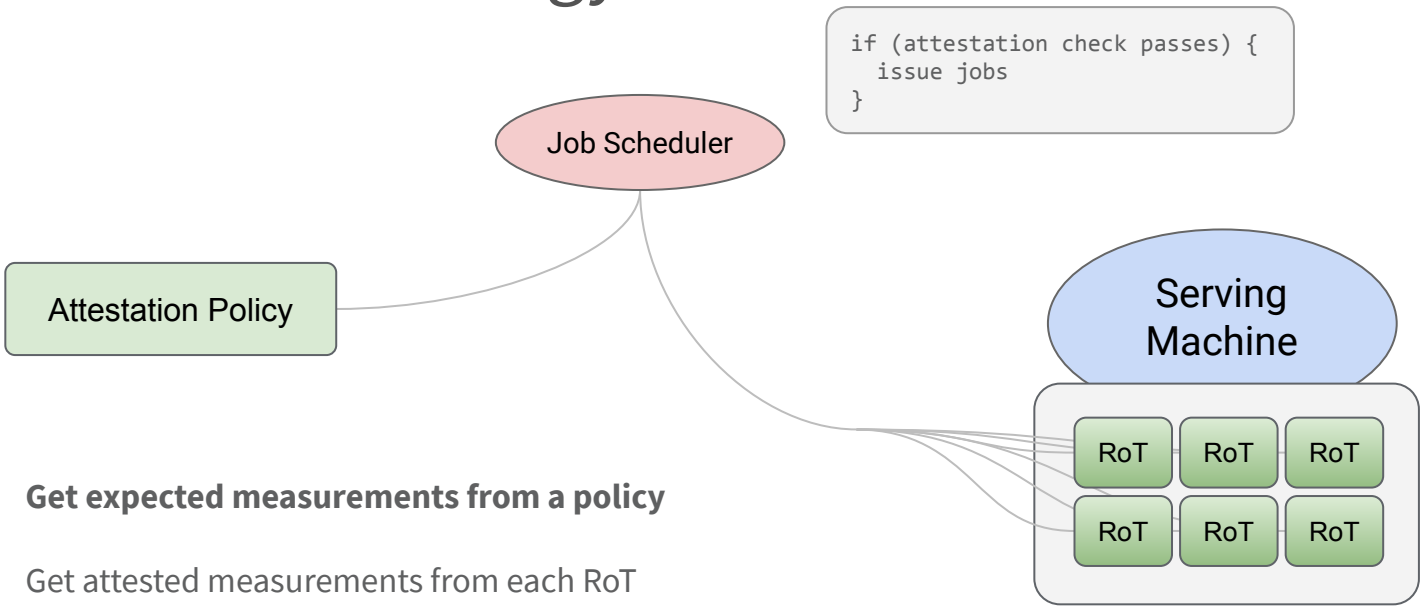
SECURITY

- Decrease machine complexity and increase reliability
  - **Very** hard to do reliable boot-time attestation on disaggregated machines
- **Centralized enforcement** allows flexible reaction
  - Control plane has wider visibility over the production fleet
- Some amount of local policy enforcement may be warranted
  - Not targeted for now, need more data on reliability at scale

Attestation failure causes



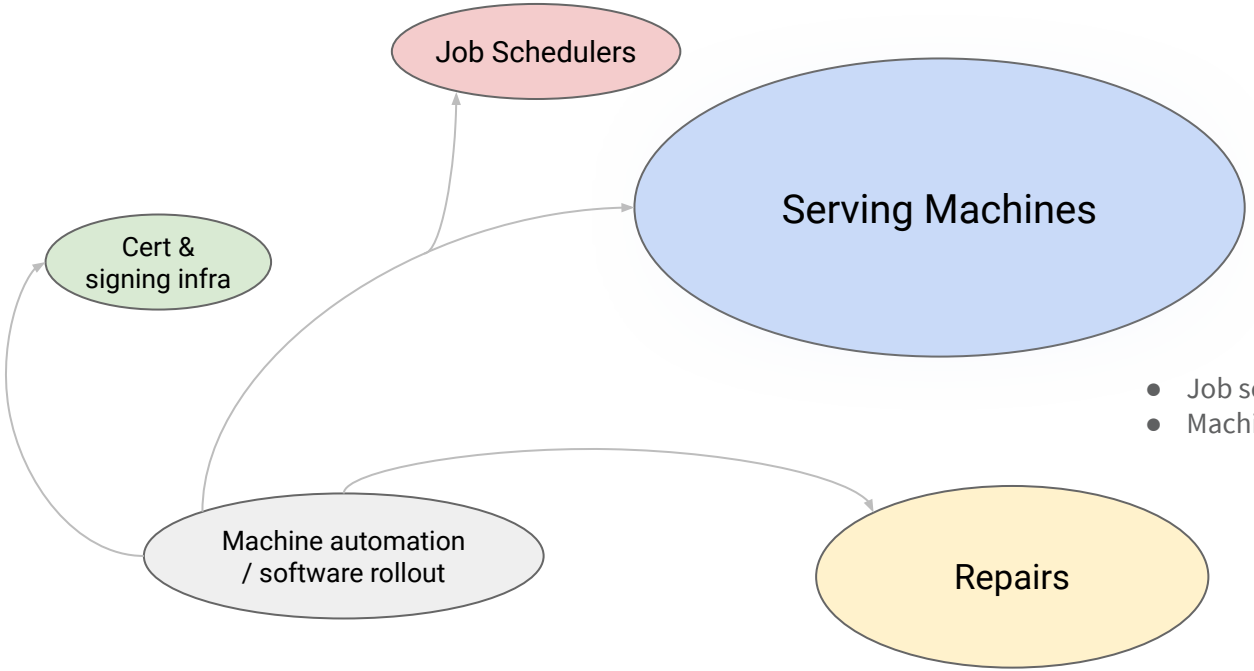
Attestation strategy



1. **Get expected measurements from a policy**
2. Get attested measurements from each RoT
3. Compare attestations to expectations

SECURITY

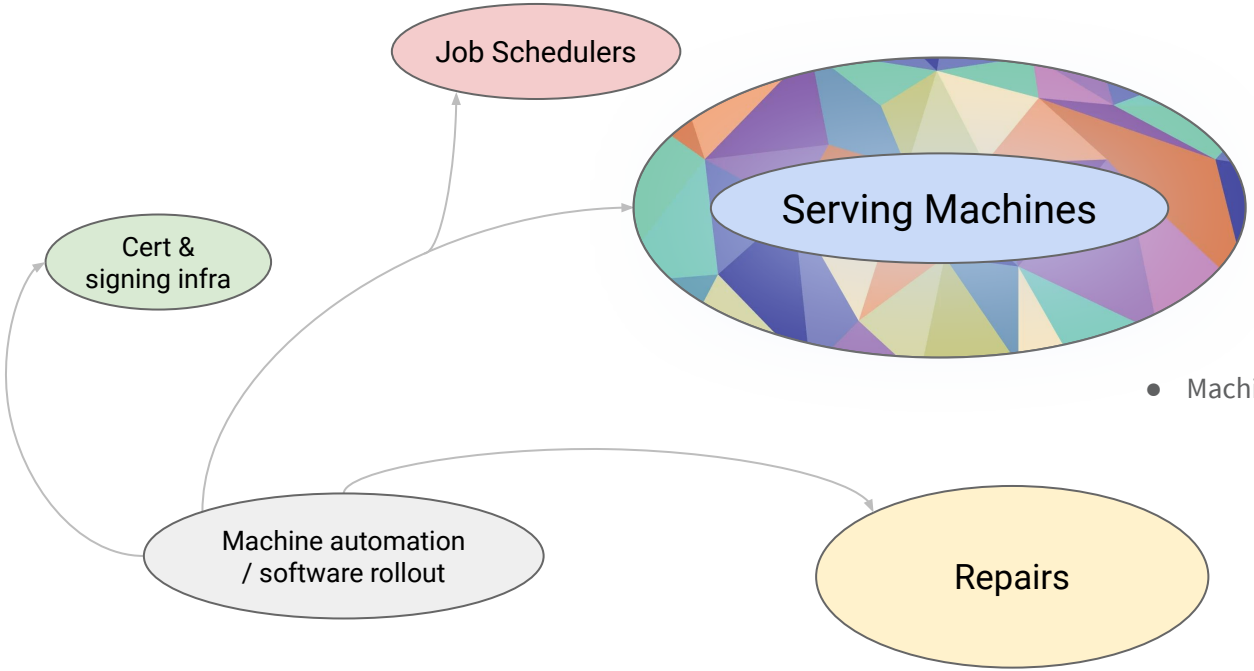
On the subject of availability



SECURITY

- Job schedulers **must** be highly available
- Machine automation can be less available

On the subject of availability



SECURITY

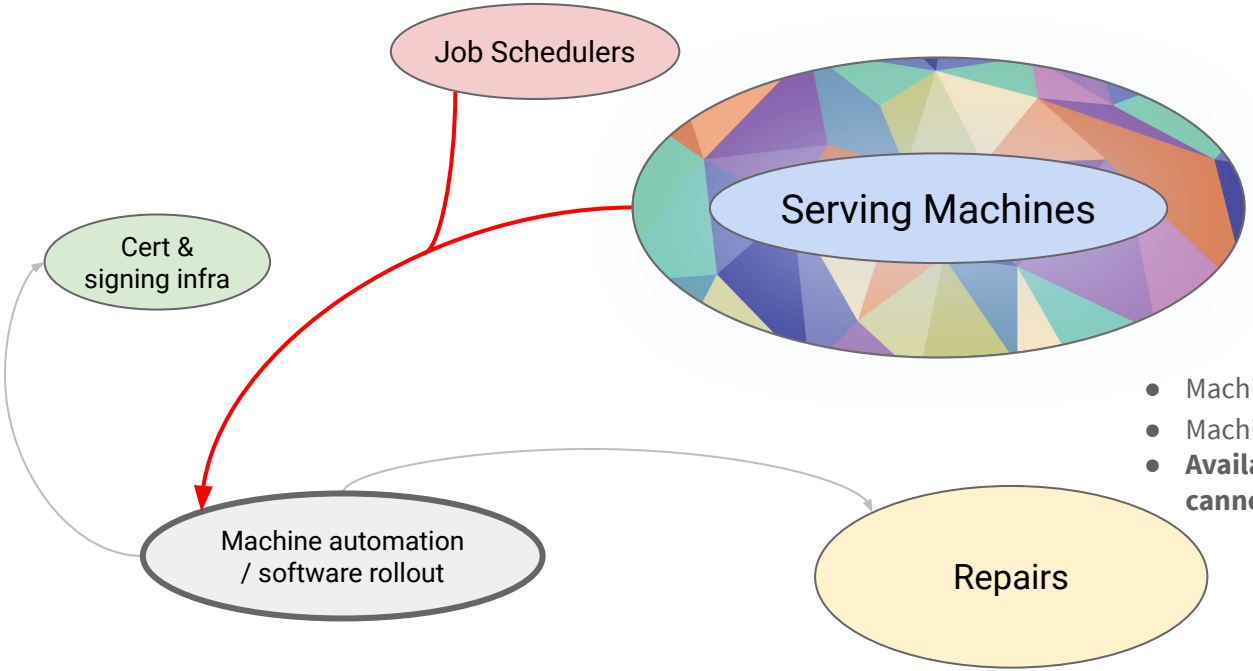
- Machines are not homogeneous

OPEN POSSIBILITIES.

Graphic from Vecteezy.com

OCP GLOBAL SUMMIT
NOVEMBER 9-10, 2021

On the subject of availability



SECURITY

- Machines are not homogeneous
- Machine automation knows intended state
- **Availability inversion! Job schedulers cannot depend on machine automation**

OPEN POSSIBILITIES.

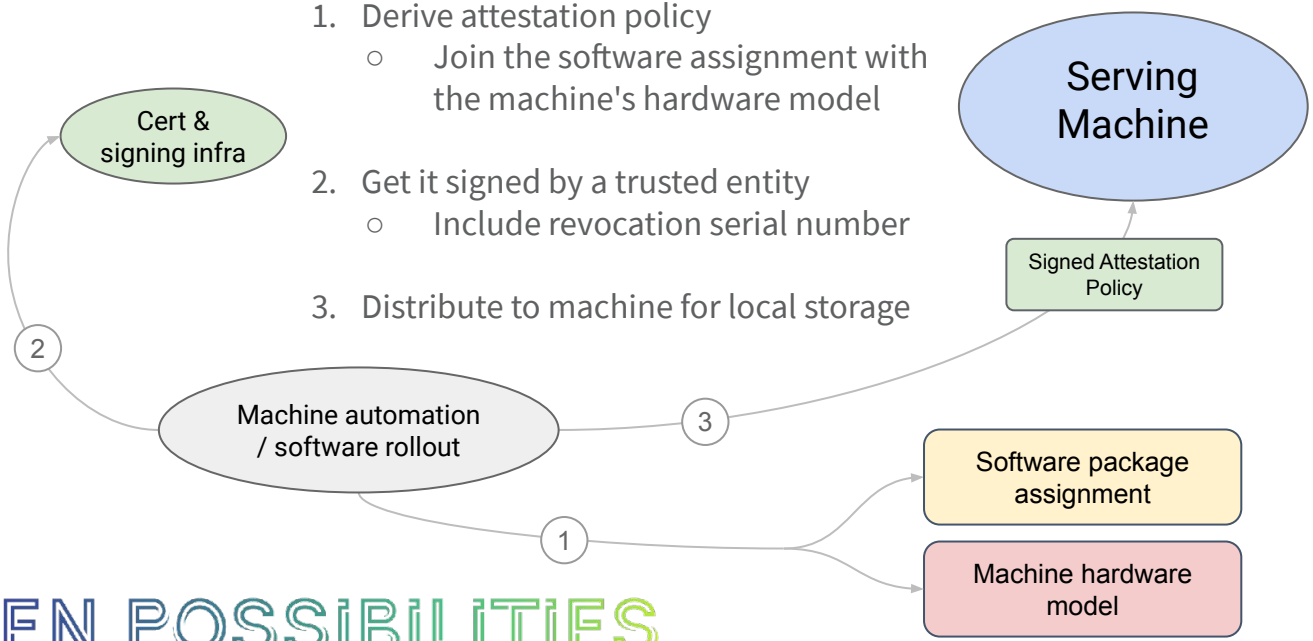
Graphic from Vecteezy.com

OCP GLOBAL SUMMIT
NOVEMBER 9-10, 2021

Store signed policies on-machine

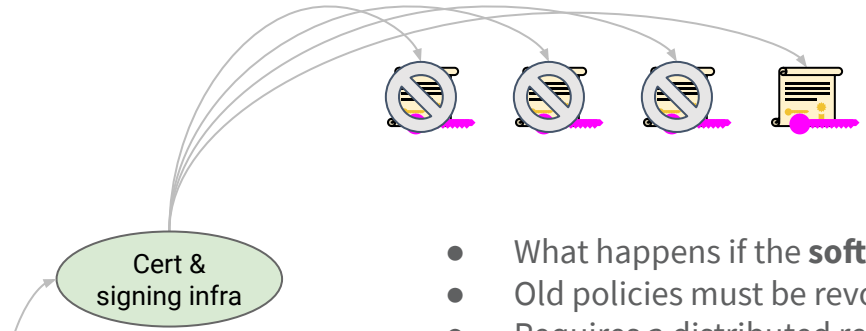
SECURITY

1. Derive attestation policy
   - Join the software assignment with the machine's hardware model
2. Get it signed by a trusted entity
   - Include revocation serial number
3. Distribute to machine for local storage



OPEN POSSIBILITIES.

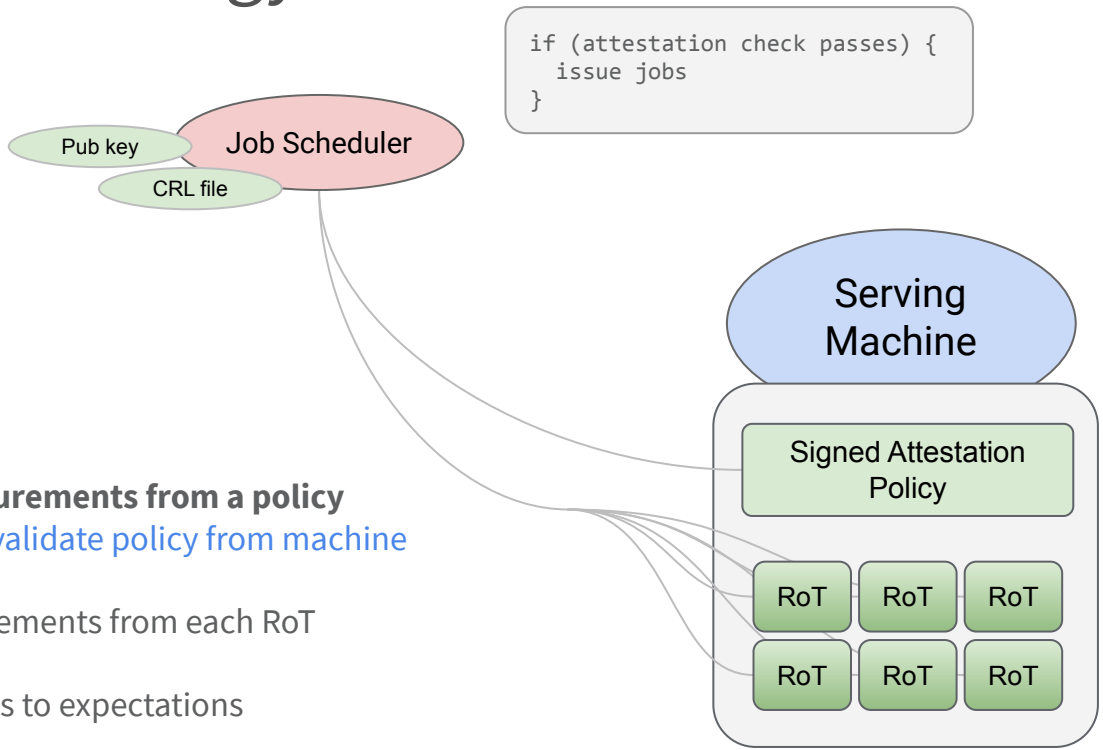
Revoking signed policies



SECURITY

- What happens if the **software intent** or **hardware model** change?
- Old policies must be revoked
- Requires a distributed revocation mechanism
  - Such as a CRL (certificate revocation list)

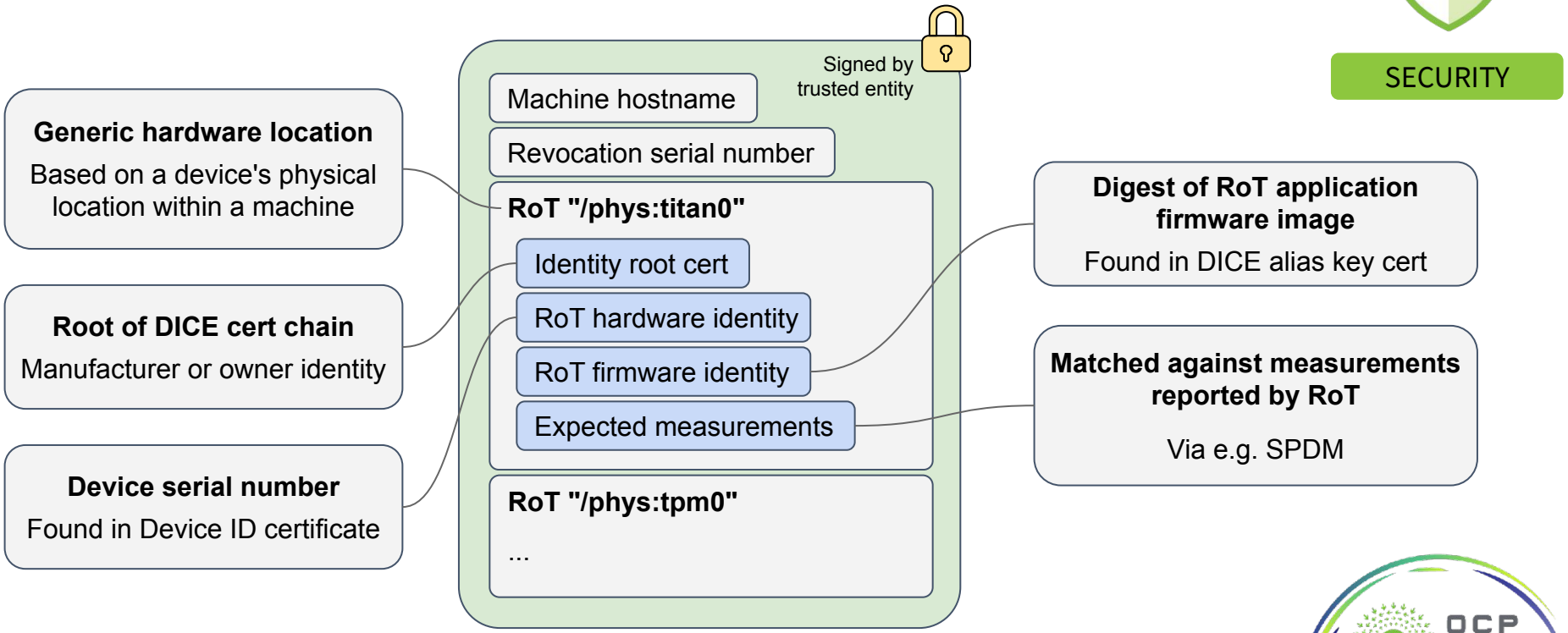
Attestation strategy



SECURITY

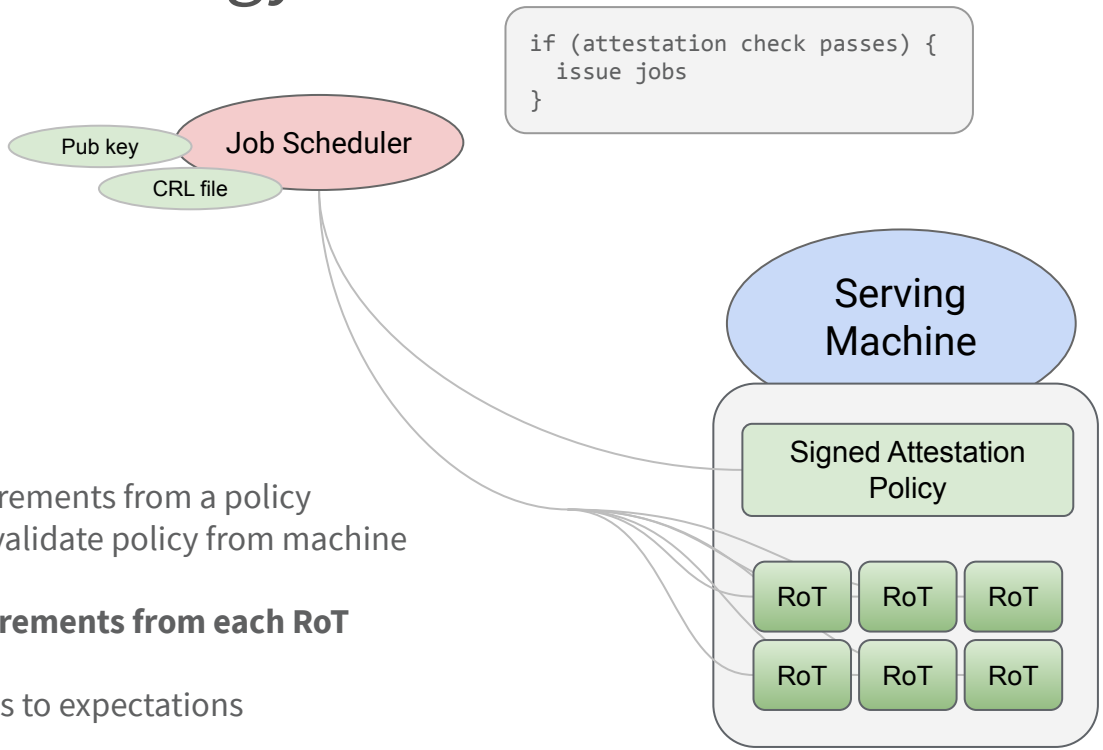
1. **Get expected measurements from a policy**
   - Retrieve and validate policy from machine
2. Get attested measurements from each RoT
3. Compare attestations to expectations

What's in the attestation policy?



SECURITY

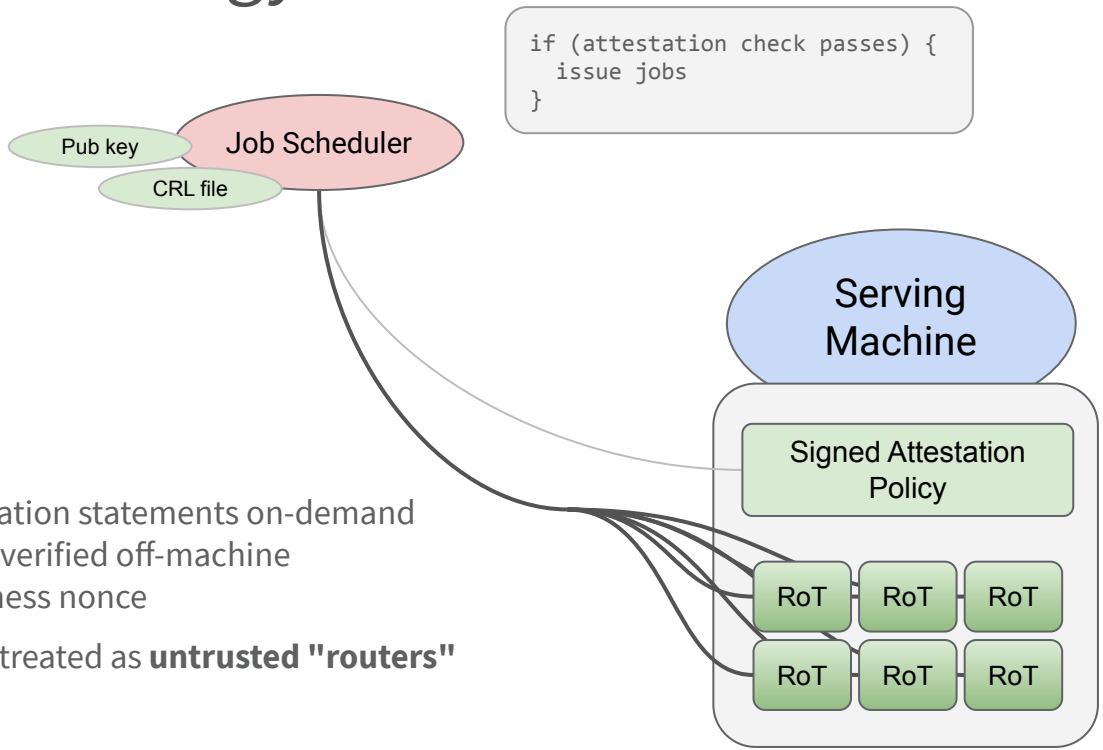
Attestation strategy



SECURITY

1. Get expected measurements from a policy
   - Retrieve and validate policy from machine
2. **Get attested measurements from each RoT**
3. Compare attestations to expectations

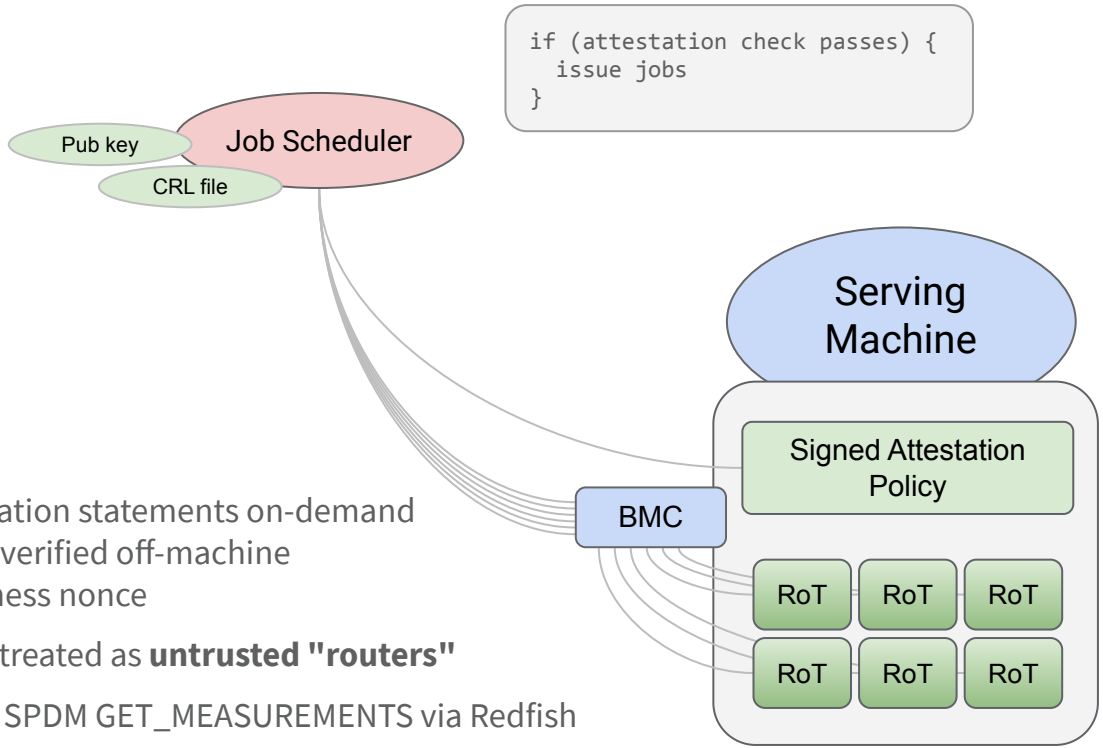
Attestation strategy



SECURITY

- Each RoT emits attestation statements on-demand
  - Signed by RoT, verified off-machine
  - Includes freshness nonce
- Intermediate entities treated as **untrusted "routers"**
  - Such as BMCs

Attestation via BMCs



SECURITY

- Each RoT emits attestation statements on-demand
  - Signed by RoT, verified off-machine
  - Includes freshness nonce
- Intermediate entities treated as **untrusted "routers"**
- One implementation: SPDM GET_MEASUREMENTS via Redfish

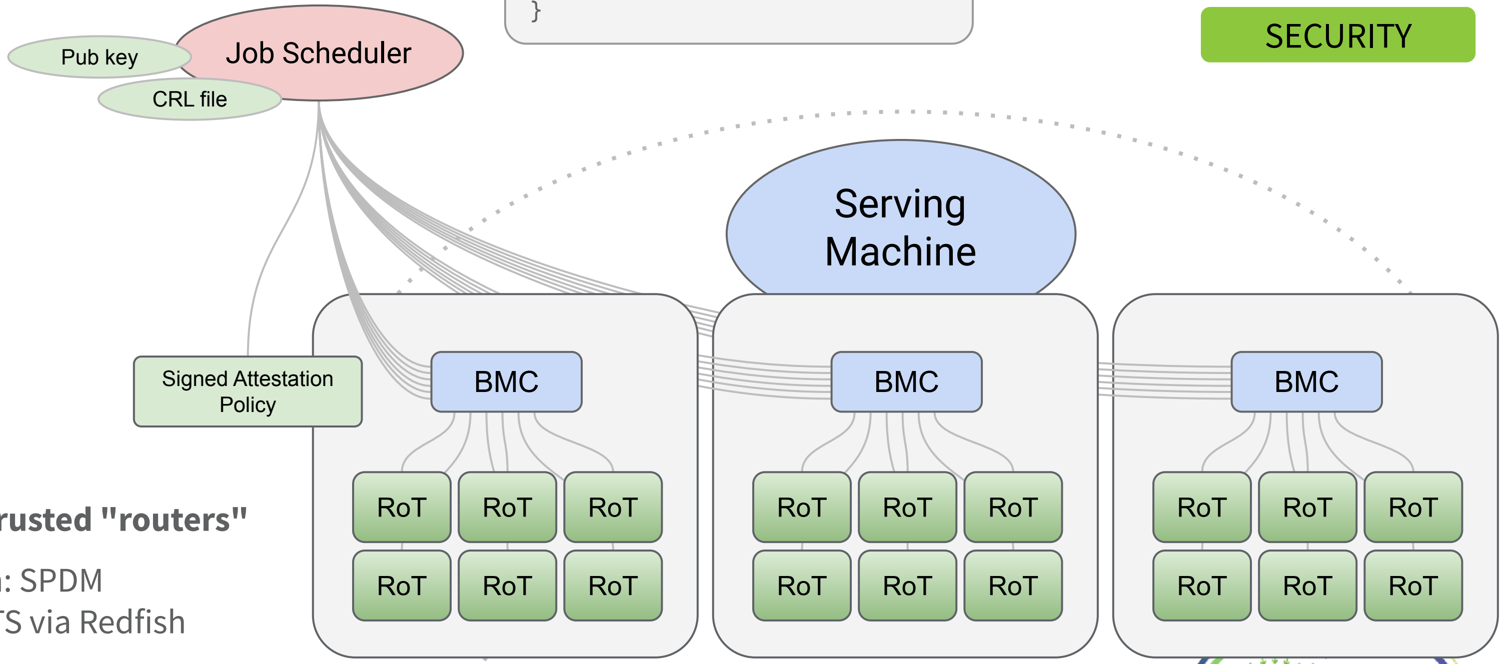
OPEN POSSIBILITIES.

BMC = Baseboard Management Controller
SPDM = Security Protocol and Data Model
Redfish = Machine management protocol

Attestation via BMCs

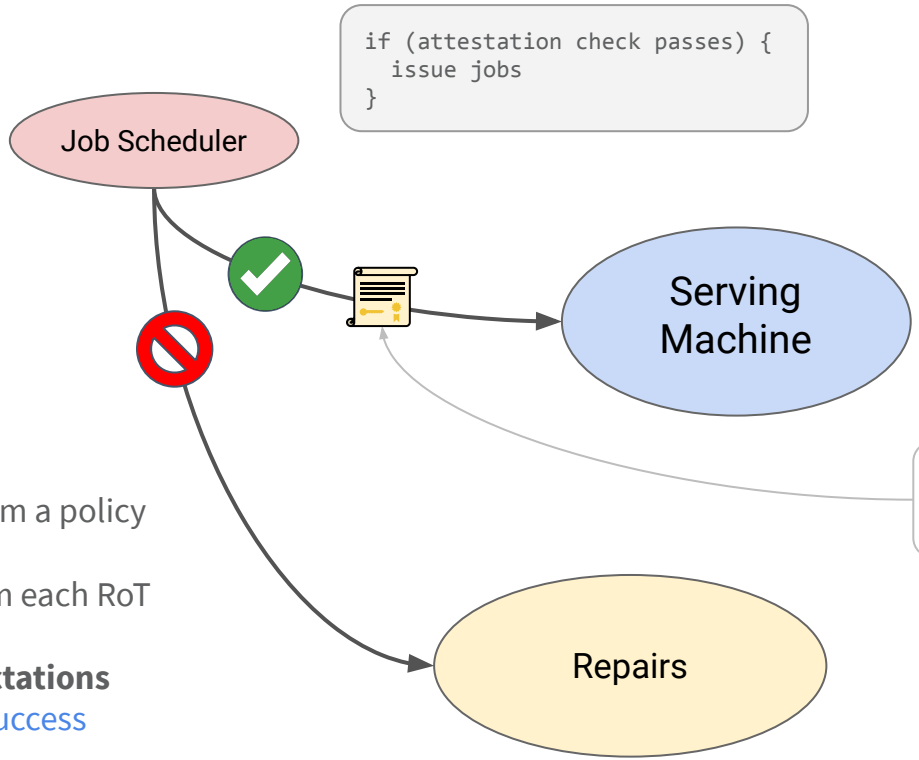
SECURITY

```
if (attestation check passes) {
  issue jobs
}
```



- BMCs treated as **untrusted "routers"**
- One implementation: SPDM GET_MEASUREMENTS via Redfish

Attestation results



SECURITY

1. Get expected measurements from a policy
2. Get attested measurements from each RoT
3. **Compare attestations to expectations**
   - Issue job + task cred on success

OPEN POSSIBILITIES.

Graphic from Emoji One

Closed vs Open

SECURITY

- Some parts of this scheme are custom (and that's okay)
  - Software packaging & rollout infra
  - Machine hardware model
  - Job scheduling infra
  - Revocation mechanisms
- Some parts can and should be open
  - Policy format
    - See CoRIM+CoMID+CoSWID
  - Exposing RoT signatures to the off-machine verifier
    - DMTF proposal: SPDM GET_MEASUREMENTS over Redfish

Call to Action

SECURITY

- **Get involved**
  - Join the OCP Security Project (& TCG/DMTF)
- **Close standards gaps, for example:**
  - Need common SPDM measurement profiles
  - Would like TPM to support "first instruction integrity"
- **Share your experiences**
  - A rising tide lifts all boat

- Keep an eye on Google's Platform Integrity Github repo: google/PINT
- Contact us: {jeffandersen, jdcooke}@google.com

OPEN POSSIBILITIES.

OCP GLOBAL SUMMIT
NOVEMBER 9-10, 2021

Thank you!