DICE Protection Environment

Ionut Mihalcea

28 March 2024

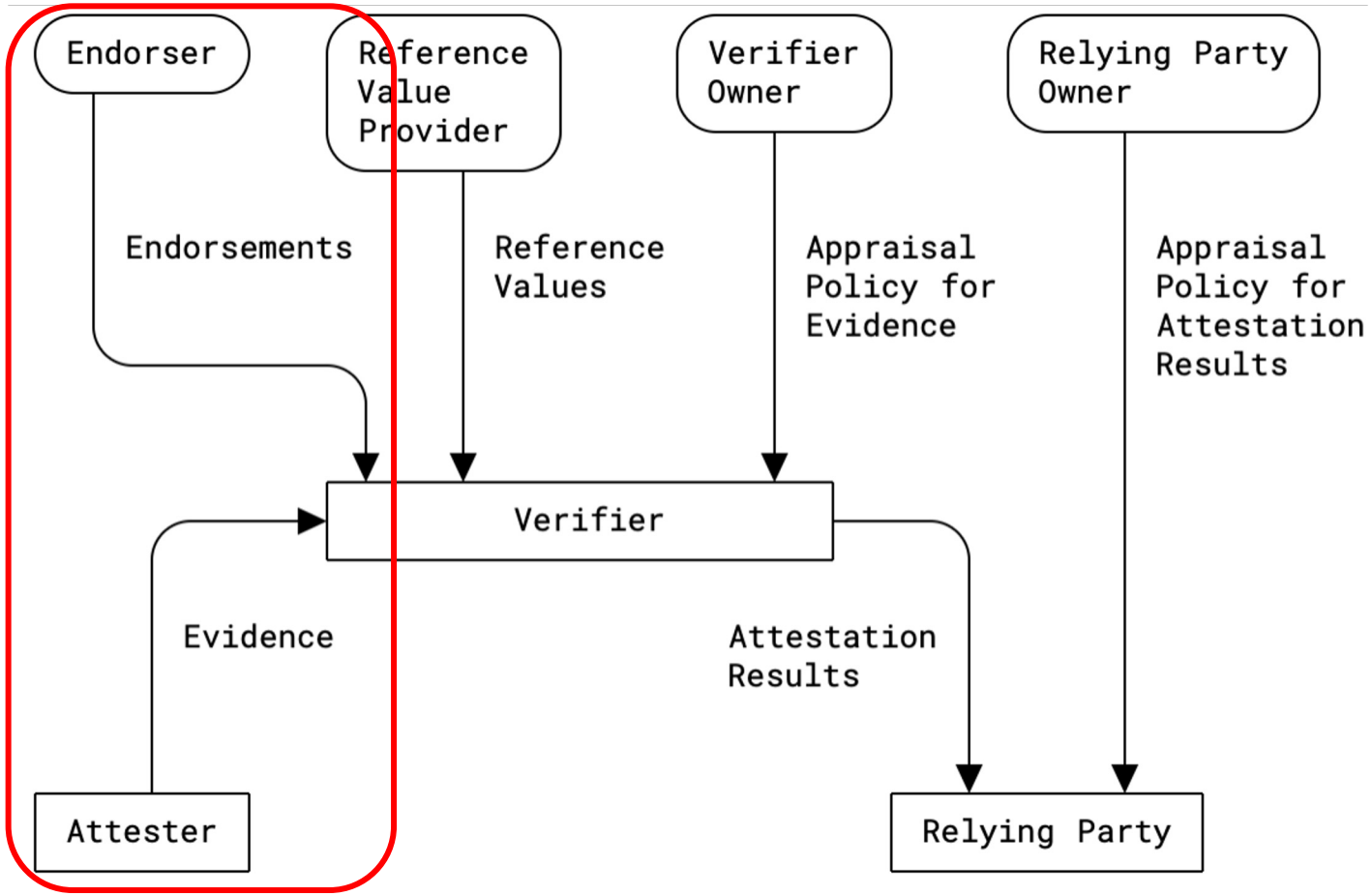
Who Am I?

- Ionuț Mihalcea
  - Software Engineer in Arm
  - Prototype systems-level code to help inform architectural design decisions

Agenda

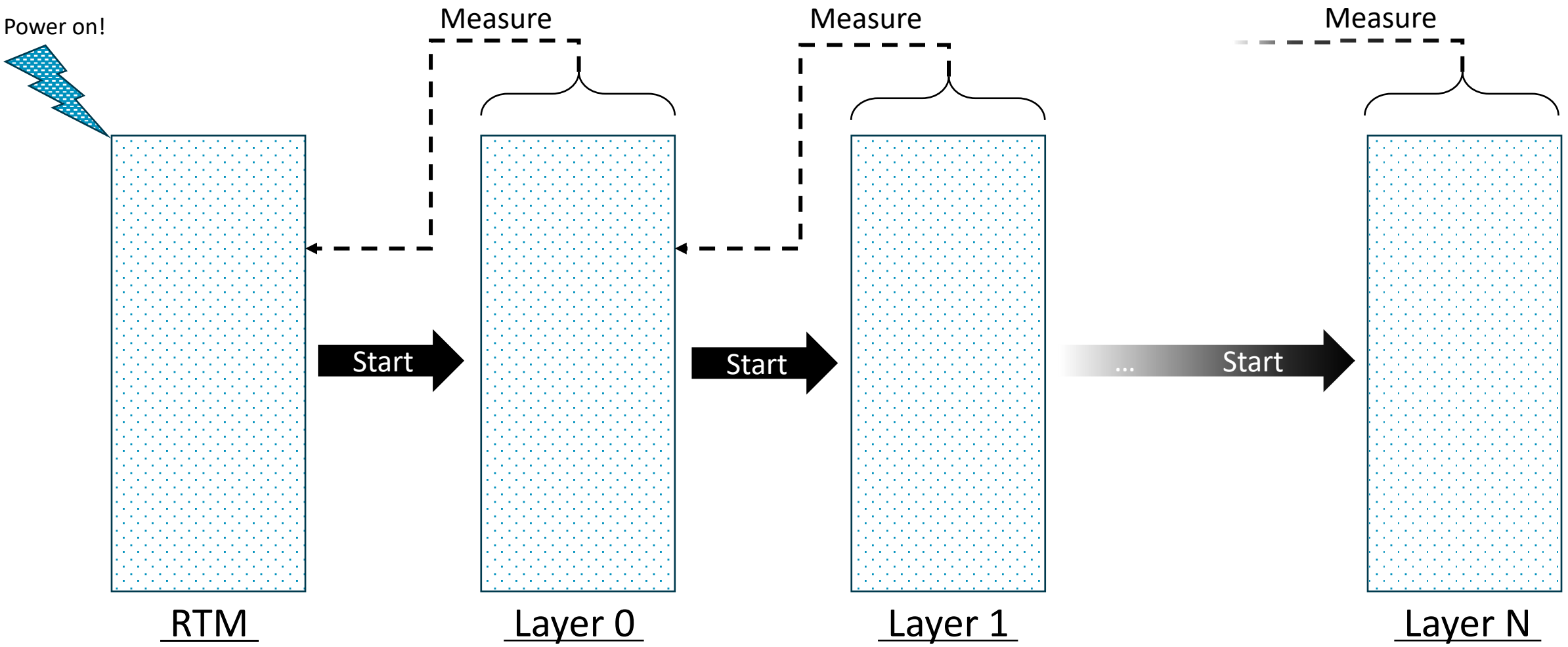
- Introduction to **R**oots of **T**rust for **M**easurement and **R**eporting
- **D**evice **I**dentifier **C**omposition **E**ngine
- **D**ICE **P**rotection **E**nvironment
- Formal Verification

RATS roadmap



RTM & RTR 101

Measured boot overview

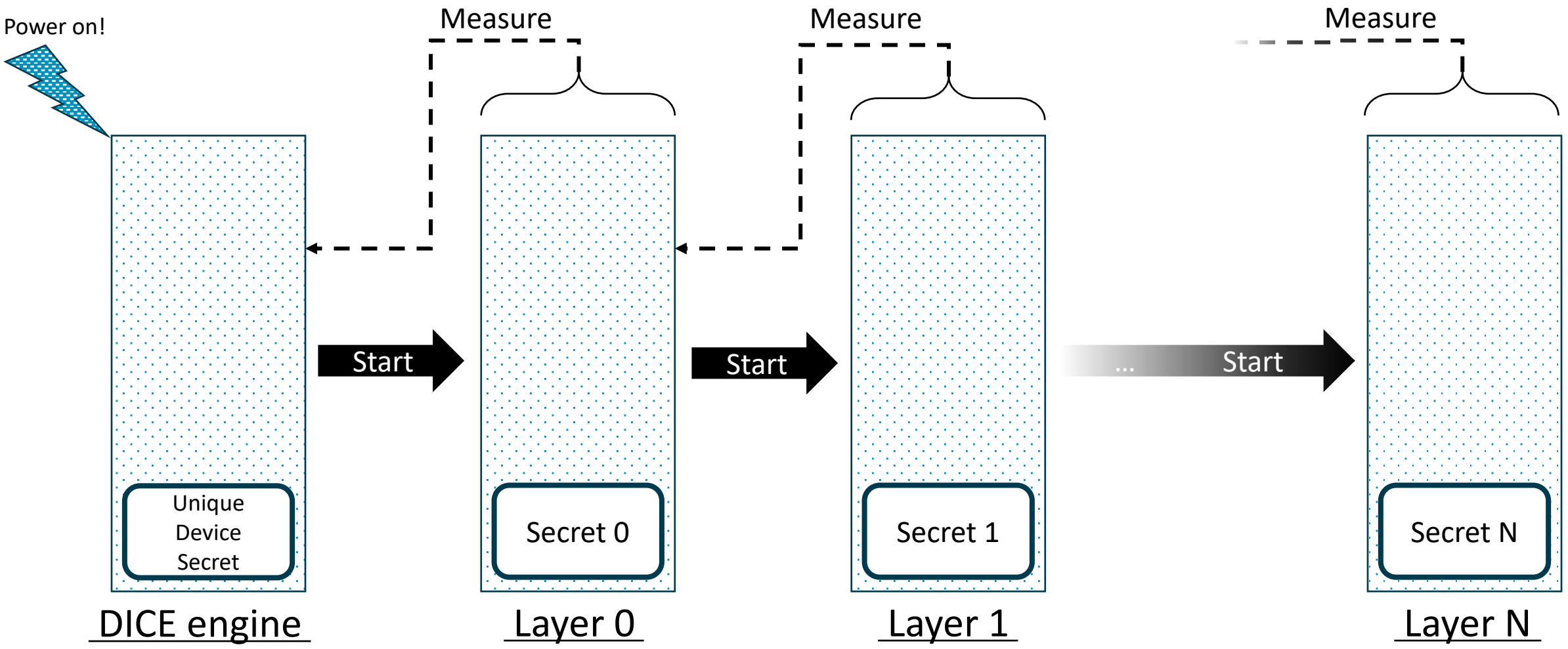


Starting a RIoT

- TPM2.0
  - Hardened component
  - Can store measurements and report on them
  - Too expensive for constrained applications (e.g., MCUs), susceptible to some physical attacks

- **R**obust, **R**esilient, **R**ecoverable **I**nternet **o**f **T**hings
  - *"Architecture for providing foundational trust services to computing devices"*, published by Microsoft
  - Provides a compact RTM + RTR for cases that can't support a TPM
  - Evolved into DICE

DICE overview

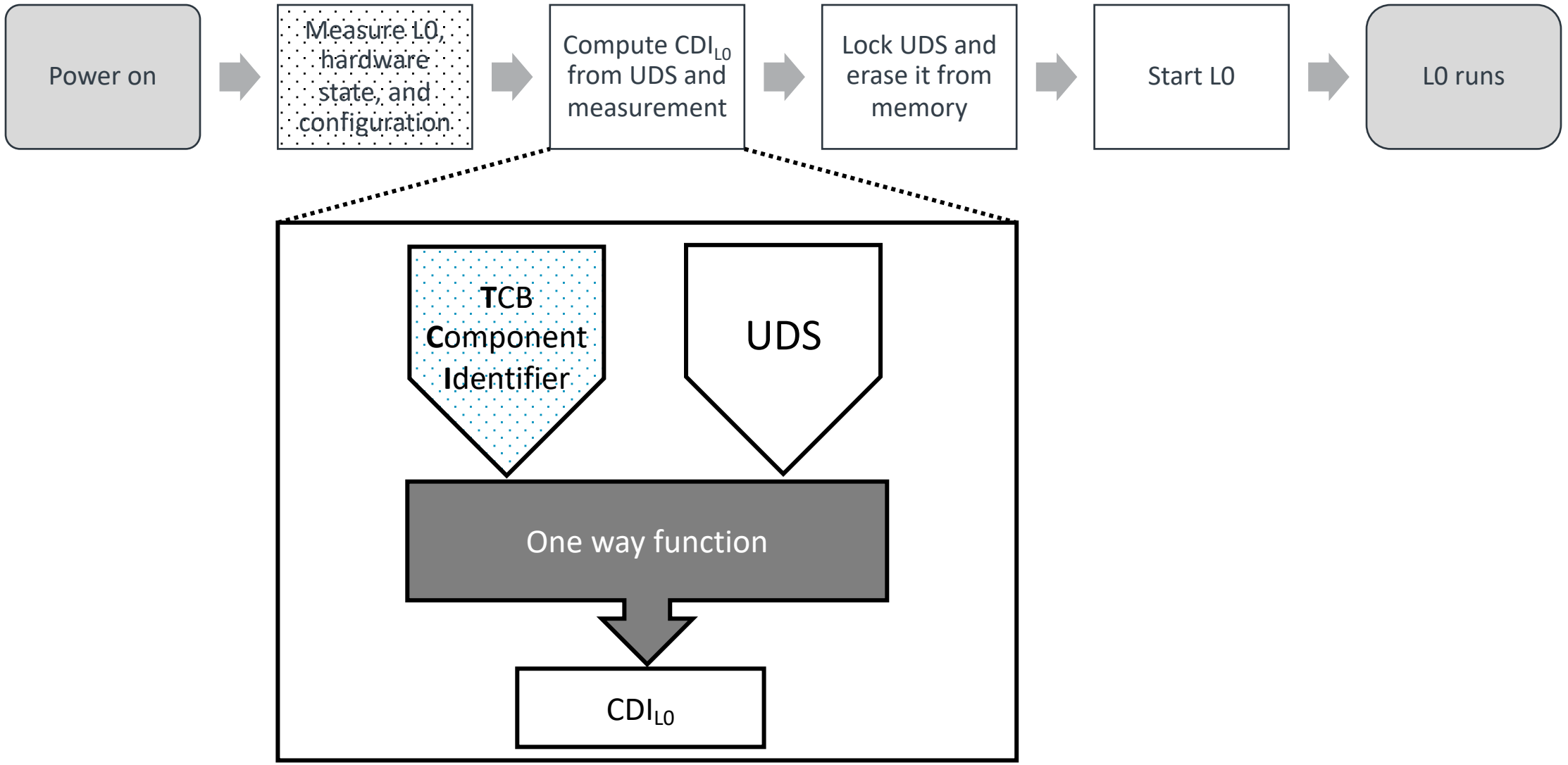


DICE primer

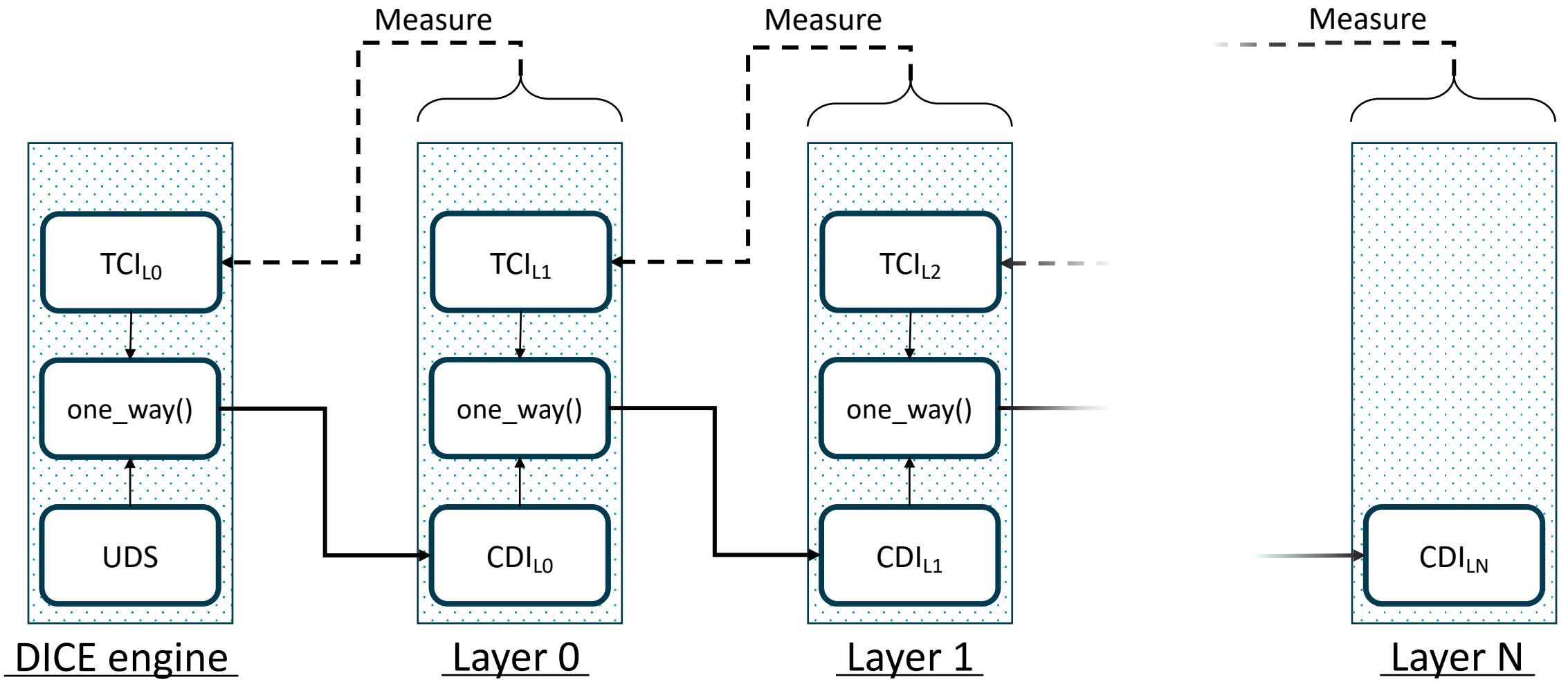
Core principles

- **U**nique **D**evice **S**ecret
  - Entropy source for the identity of all layers
  - Provisioned at manufacturing time
- **C**ompound **D**evice **I**dentifier
  - Recursively derived from UDS, measurements, and other metadata
  - Uniquely identifies a component in the stack, including its ancestry
- Layering
  - Each layer gets its own credentials
  - The credentials are derived recursively from the UDS

Root of Trust



Layering


Measure
Measure
Measure
TCI$_{L0}$
TCI$_{L1}$
TCI$_{L2}$
one_way()
one_way()
one_way()
UDS
CDI$_{L0}$
CDI$_{L1}$
CDI$_{LN}$
DICE engine
Layer 0
Layer 1
Layer N

Benefits

- Compact RoT
- Strong device identity
- Integrated certificate chains
- Both explicit and implicit attestation
- No requirement for durable storage beyond UDS

Challenges

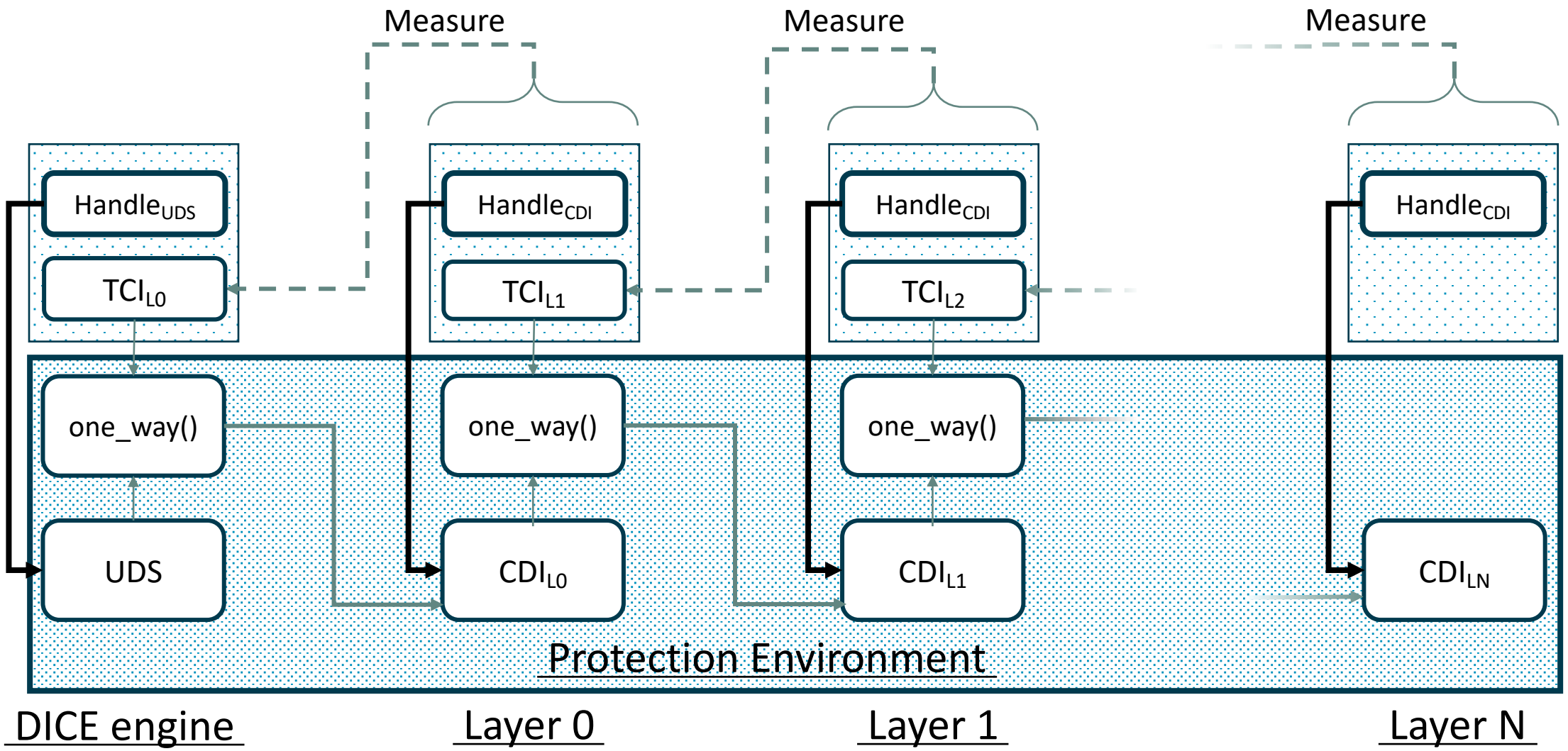
- CDI handling / protection
- Performance
- Interoperability & consistency
- Sealing

DICE Protection Environment primer

Purpose

- DPE is a new architectural component that offers:
  - An interface for using and managing DICE component contexts
  - Protection for the DICE secrets of each component
  - Enforcement of DICE-related policies

Layering with DPE



DPE benefits

- Large part of DICE implementation offloaded to DPE
- Secure handling and caching of secrets
- Expensive cryptographic operations offloaded to DPE

Formal verification

DICE*

- Formally-verified implementation of DICE measured boot
- Implements the DICE Engine and L0 components
- Verification goals:
  - Confidentiality
  - Functional correctness
  - Memory safety
  - Side-channel resistance

DICE*: Implementation details

- Implemented in Low*
- Split into platform-agnostic and platform-specific parts
- Relies on HACL* for the cryptography
- Relies on a (new) verified library for (a subset of) ASN.1 and X.509

- The verified implementation shows no performance impact compared to an unverified version!

Looking forward…

- DPE is still in the process of standardization
  - … which could benefit from formal modelling to iron out any issues
- For use in the field, a formally verified implementation should not sacrifice readability and maintainability
  - Is this possible with current tools and frameworks?

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు

Bibliography

- RIoT: https://github.com/microsoft/RIoT
- TCG DICE: https://trustedcomputinggroup.org/work-groups/dice-architectures/
- TCG DPE (work in progress): https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Protection-Environment-Version-1.0-Revision-13_21March24.pdf
- DICE* paper: https://www.usenix.org/system/files/sec21-tao.pdf
- DICE* implementation: https://github.com/verified-HRoT/dice-star