2017 International Trusted Computing and Security Innovation Summit
April 6, 2017, Beijing
Session III: Trusted Computing Techniques

Trusted Computing and Securing Devices

Rob Spiger, Principal Security Strategist, Microsoft
Rob Spiger，首席安全策略师，TCG董事，微软
Stefan Thom, Principal Engineer, Windows Division, Microsoft (TCG Promoter Member Company)
Stefan Thom, 首席工程师，Windows部门，微软（TCG董事成员）

Introduction

- The cloud today is connected with numerous devices
- Security continues to evolve to address threats

Device Security Techniques

- Traditional: PCs with TPM
  - Proven trusted computing benefits

- Recent: IoT Devices with TPMs
  - Fabulous when practical

- IoT Devices with DICE
  - Something new in draft from TCG
  - Device Identifier Composition Engine (DICE)

DICE Goals

- Improve security for updatable boot code
- Supports low power and low cost devices
- Identify the device and its software when connecting to cloud services
- Protect data by preventing access by old versions of software
- Cryptographically agile

Technology Timeline

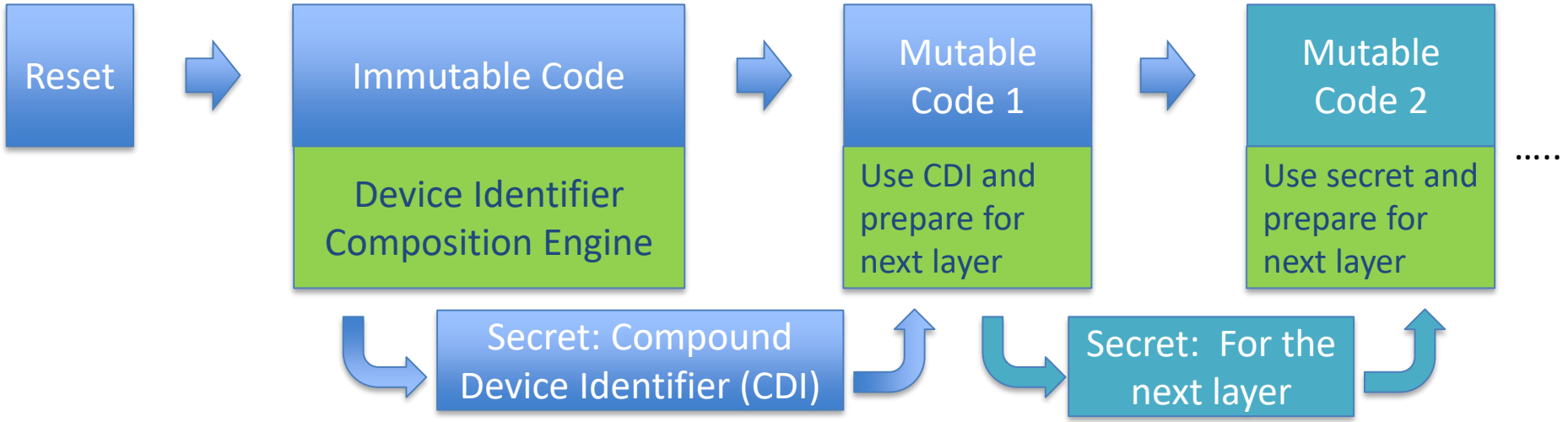
- April 2016: Microsoft Research Robust Internet of Things (RIoT) Paper
- October 2016: TCG forms the DICE Architectures working group
- December 2016: TCG shares a draft DICE specification for public comments
- Ongoing: Microsoft publishes example code

DICE Basics

- Security is divided into boot layers
- Starts with unconditional actions in immutable code
- Secrets are passed to each layer of the boot process
- There is no persistent isolated environment
  - (Major difference from a TPM)
- Security components are intended to be combined with the device ROM and firmware by the manufacturer

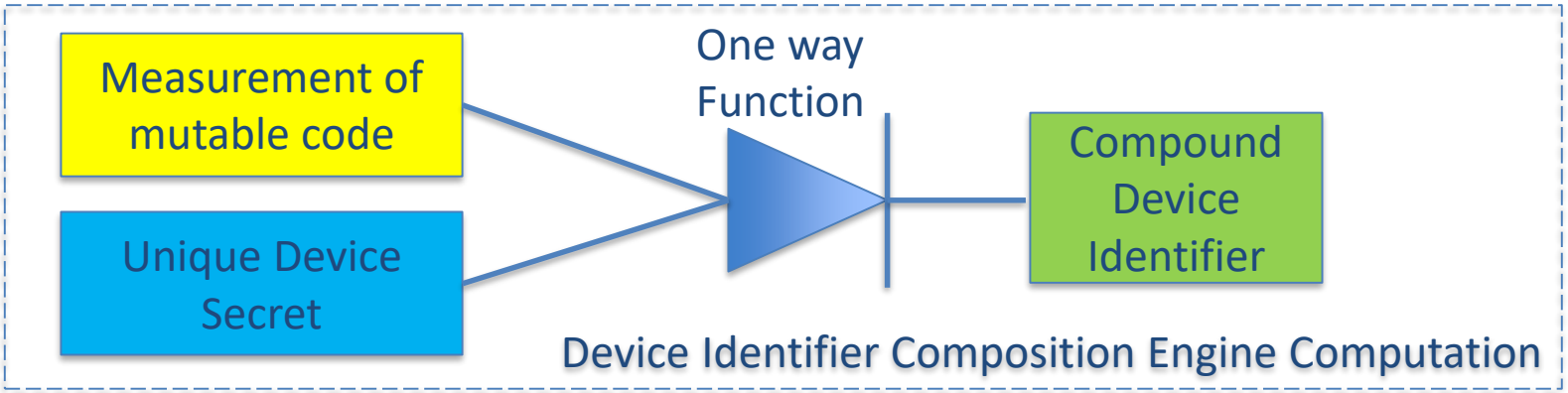
Device Identifier Composition Engine (DICE)

- Platform reset starts in a trusted state
- Engine computes a value based on mutable code



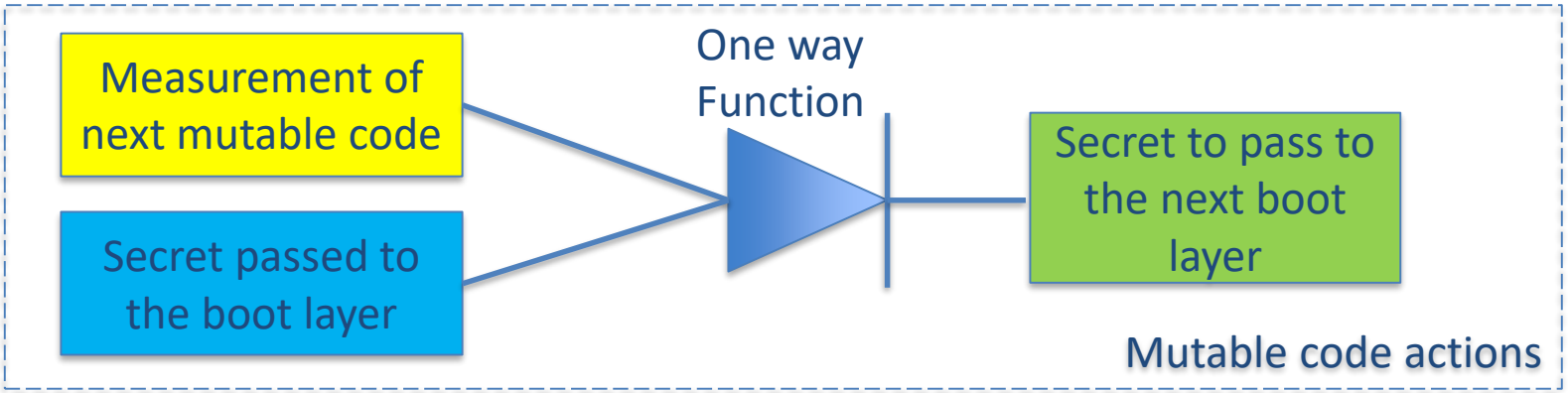
Generating the Compound Device Identifier

- Immutable code has access to the Unique Device Secret, but the mutable code does not
- Immutable code only passes the Compound Device Identifier to the first mutable code



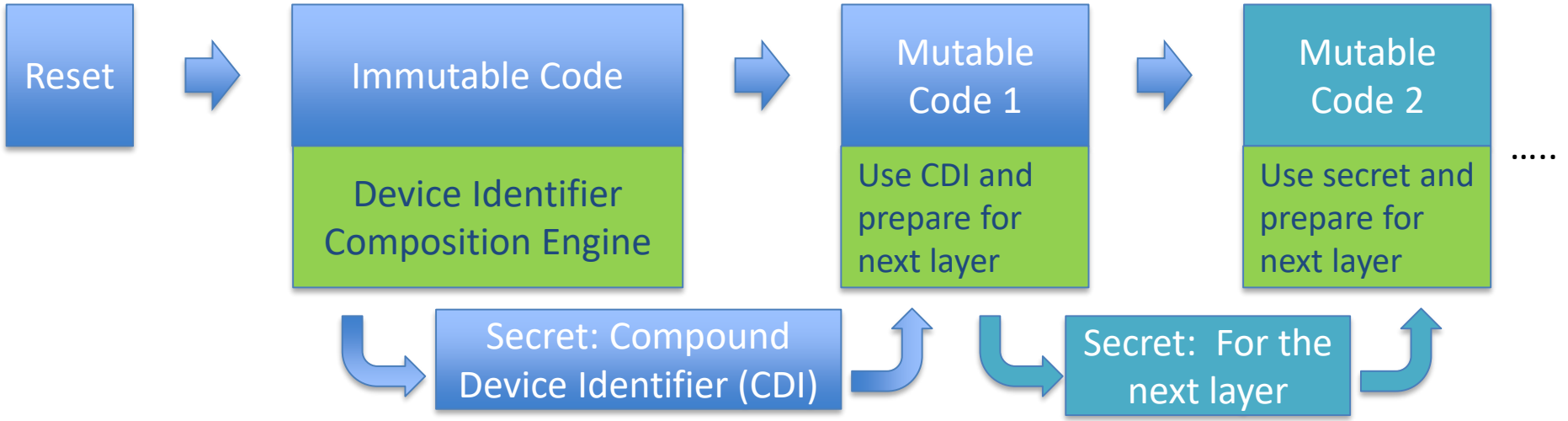
Mutable Code Actions

- Each layer of mutable code receives a secret
- The secret can be used to prove the identity and software booted
- Secret is combined with a measurement of the next boot layer



DICE Boot Flow Revisited

- The secret at each layer depends on the device identity and the code (including lower layers)



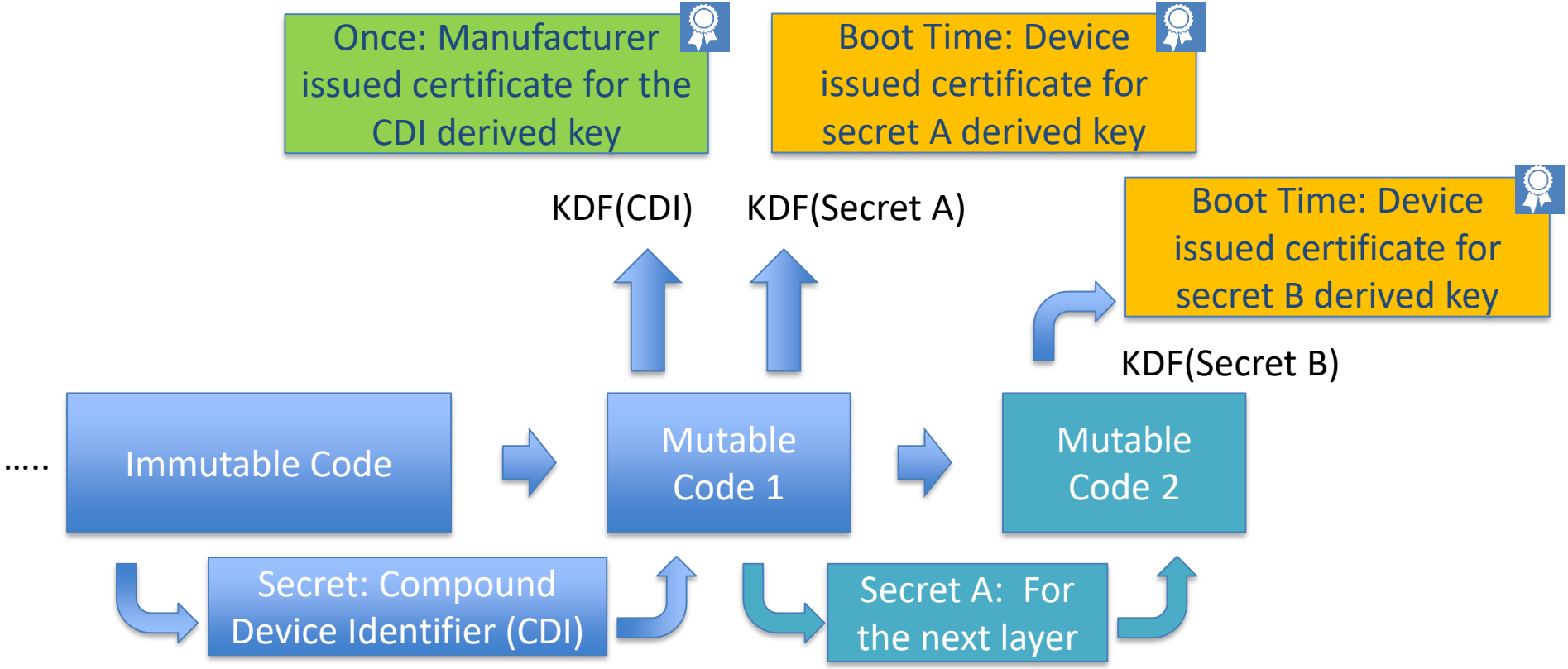
Proving device identity and software

- The CDI can be used with a key derivation function to make an asymmetric key with a public and private portion
- If the first mutable code never changes, the CDI and the derived key values will be stable
- For the first layer, the manufacturer can issue a certificate for each device it makes and include the public portion of the key
- When the device performs a computation using the private portion of the key, it proves its identity and boot software

Proving software for later in boot

- The first layer of software has a certificate from the manufacturer.
- The each layer of software can use its private key and certificate to issue a certificates for the next layer
- Each successive layer can do the same thing, creating a chain of certificates
- The certificate chain can be used to establish TLS sessions proving the device and software

Chained Certificate Issuance



MiniDICE Prototype Overview

- Open source example code implementing draft DICE requirements
- Tooling to help with common development and manufacturing tasks
- Hardware: STMicroelectronics M0+ Cortex microcontroller
- Shows how adaptable DICE concepts are for different platforms

MiniDICE Prototype Tooling

- Supports common Device Firmware Update (DFU) tool for application updates
- Tooling helps create application versions and signature blocks that include the identity of the previous version
- Helps manufacturers lock down the device by provisioning the signing key required to update the application payload later
- Shows an example of how the manufacturer might sign a device certificate

MiniDICE Prototype ROM Code

- Includes both the DICE Engine and the first layer of “mutable code” in ROM so they cannot change
- Authenticates updates and boot application
- Supports data migration by passing the CDI for current and the previous application payload
- Prevents rollback by storing and checking the application signature timestamp

Demo

- Initial ROM code and device identity already provisioned
- Build, sign and deploy an application payload
- Show the device identity secret
- Show the calculation of the compound identity
- Show how a certificate is made to prove the device's application code measurement

Conclusion

- DICE shows promise as a new security technology
- Participate by
  - reviewing publications from Microsoft about RIoT
  - reviewing publications about DICE from TCG
  - joining TCG and working with other stakeholders
  - using published samples and adapting the features to your requirements

Resources

- Microsoft Research Robust Internet of Things (RIoT) Paper: https://www.microsoft.com/en-us/research/publication/riot-a-foundation-for-trust-in-the-internet-of-things/
- TCG draft DICE specification: https://members.trustedcomputinggroup.org/kws/draft_specs/161012_Device_Identifier_Composition_Engine_Member_IP_and_Technical_Review.pdf
- MiniDICE reference code: https://github.com/LordOfDorks/miniDICE