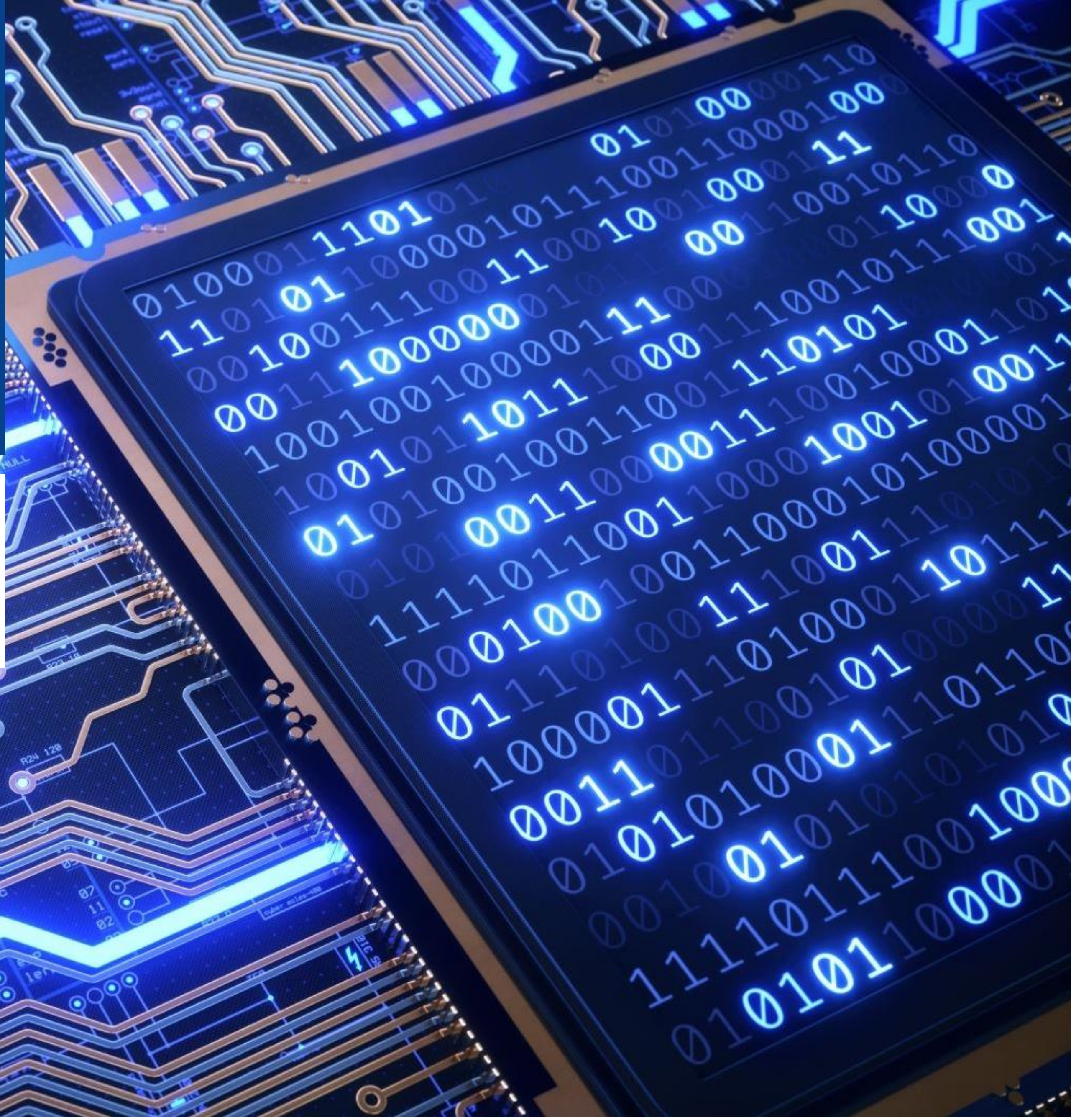
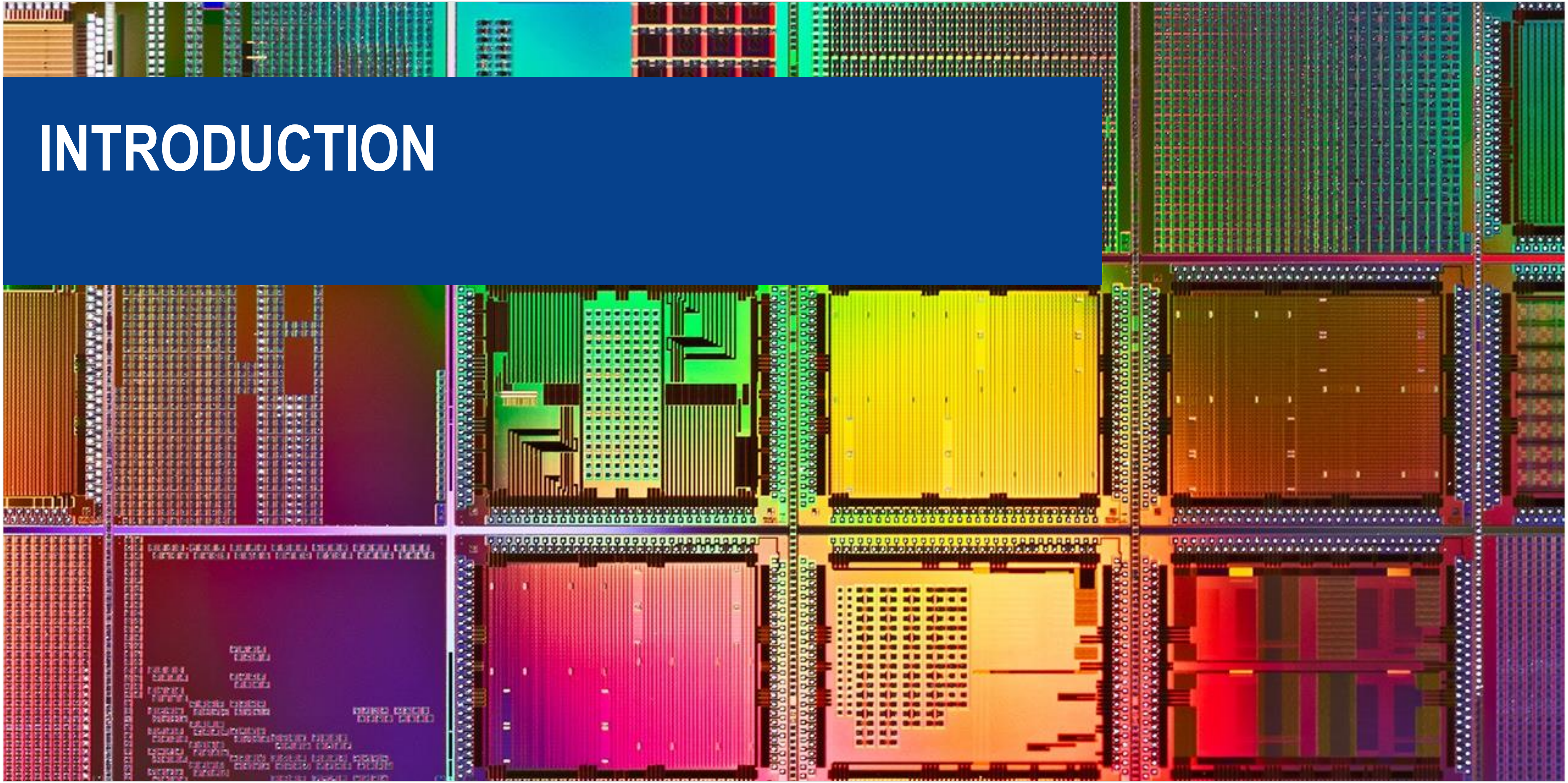
REALIZING SECURE BOOT ON AUTOMOTIVE DEVICES

DECEMBER 04, 2024

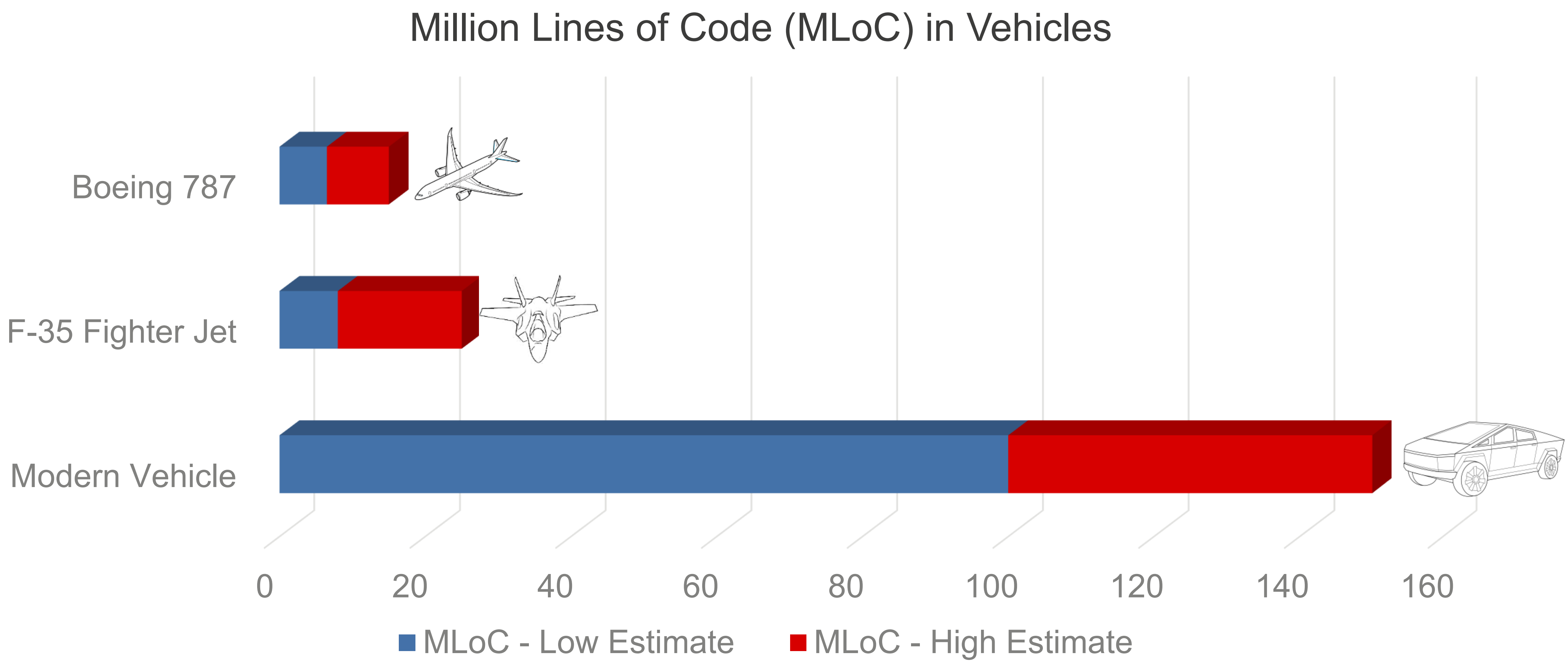
PHILIP LAPCZYNSKI
RENESAS ELECTRONICS CORPORATION



INTRODUCTION



A GROWING CHALLENGE



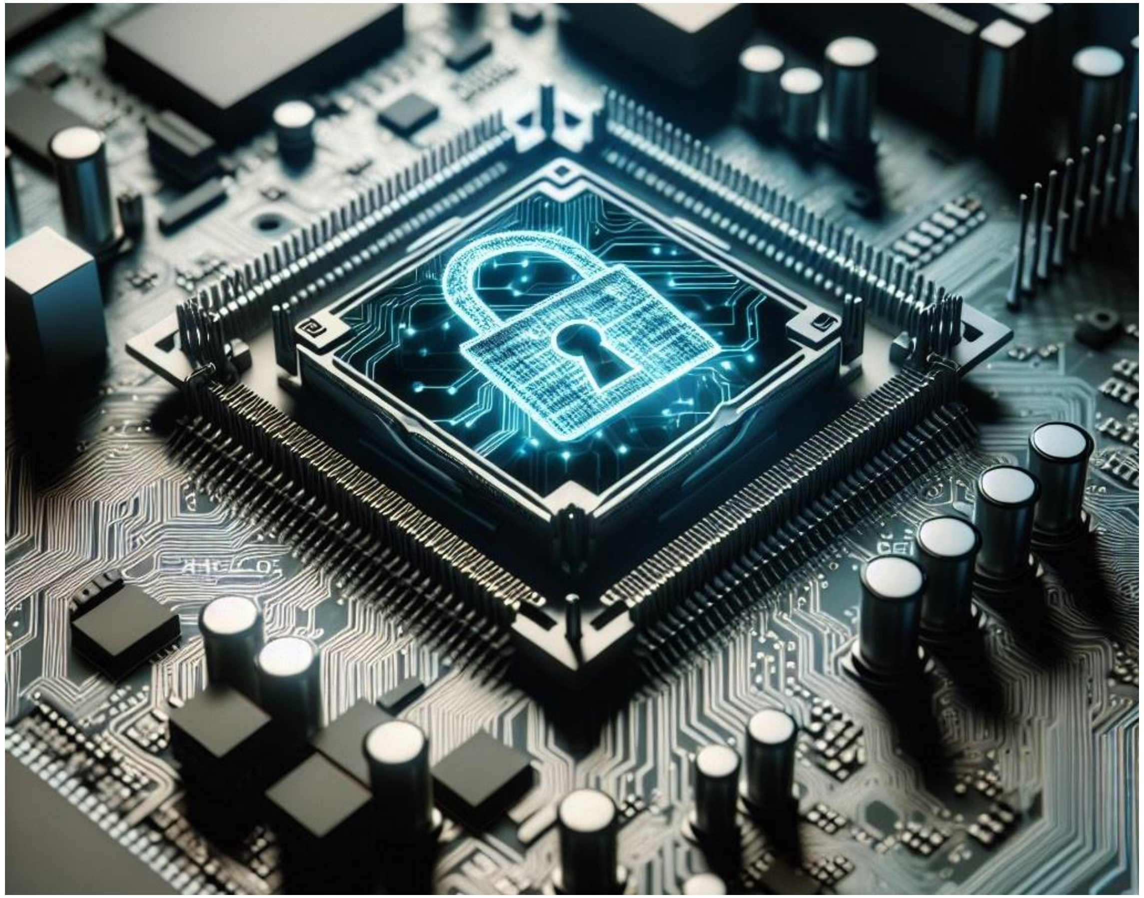
WHY SECURE BOOT?

Security Goal:

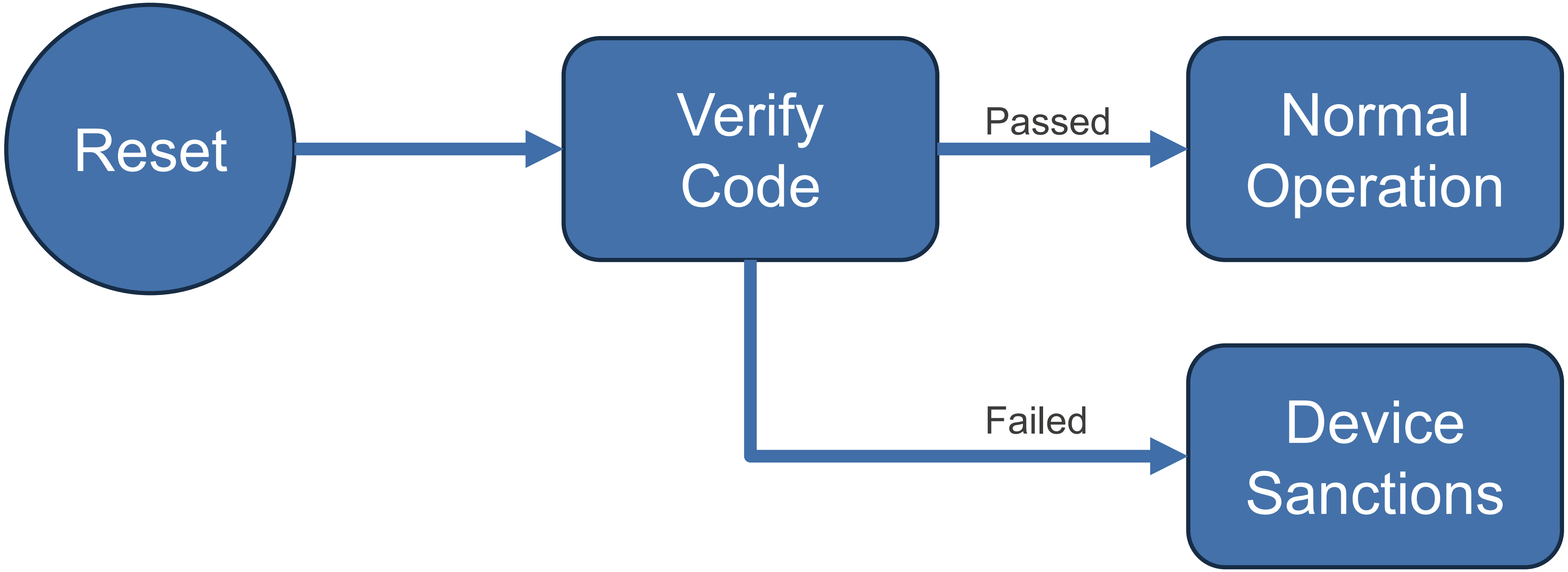
- Detect and prevent unauthorized code execution on boot
- Establish device root of trust

Security Solution:

- Check code integrity and authenticity on startup
  - Protects the device against **unauthorized software execution and tampering**
  - Support loading executable code **encrypted at rest**



SECURE BOOT INTRODUCTION - HIGH LEVEL FLOW



If secure boot fails sanctions could include:

- Disallowing access to cryptographic keys or peripherals,
- Resetting the CPU
- Executing a fallback or device recovery program.

SECURE BOOT INTRODUCTION

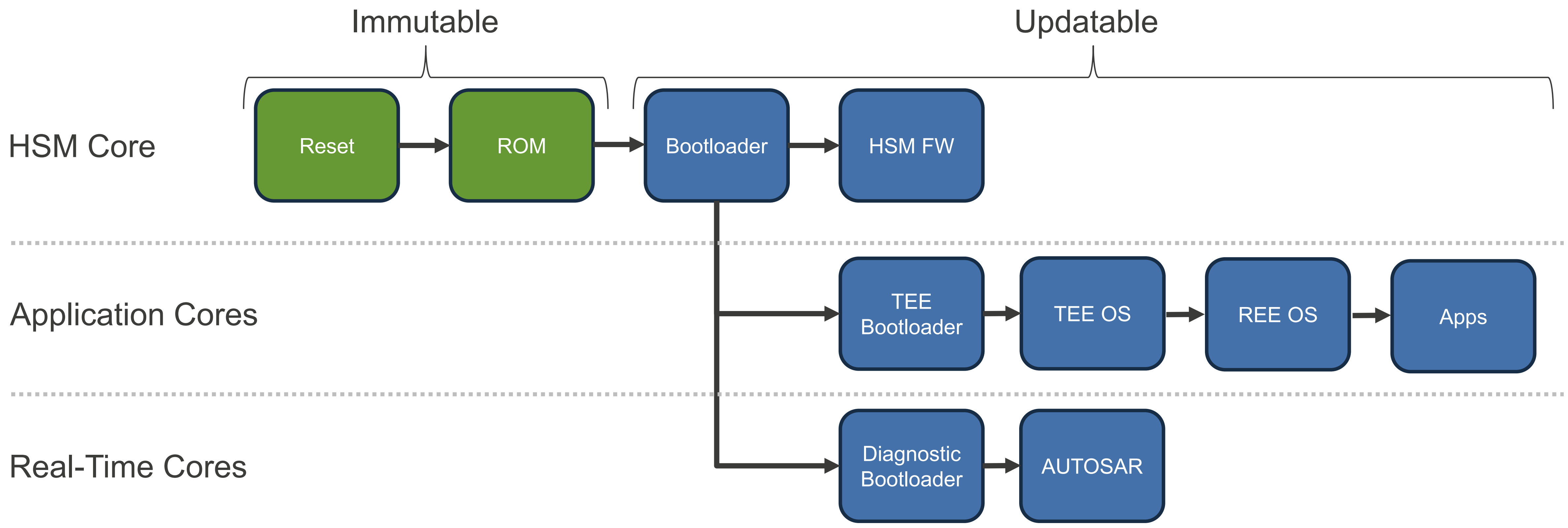
Simple, right?

SECURE BOOT OVERVIEW

SECURE BOOT OVERVIEW – WHAT IT LOOKS LIKE IN THEORY

Basic Secure Boot Example

SECURE BOOT OVERVIEW – WHAT IT LOOKS LIKE IN PRACTICE



Example secure boot on multicore SoC

SECURE BOOT OVERVIEW – HW ROOT OF TRUST PROPERTIES (1/2)

1. Immutability

- **Definition:** The RoT must be unchangeable after manufacturing or initial configuration.
- **Why It Matters:** Prevents attackers from modifying the RoT to compromise the system.

2. Minimal scope

- **Definition:** The RoT should be small, containing only the essential code or functionality required for its role.
- **Why It Matters:** Reduces the attack surface and makes security verification more feasible.

3. Integrity/Authenticity

- **Definition:** The RoT must ensure that the code and data it protects is authentic and has not been altered.
- **Why It Matters:** Preserves trust in the system's boot process and secure operations.

4. Confidentiality

- **Definition:** The RoT must safeguard sensitive data, such as cryptographic keys, from unauthorized access.
- **Why It Matters:** Prevents data leaks that could compromise the trust chain.

SECURE BOOT OVERVIEW – HW ROOT OF TRUST PROPERTIES (2/2)

5. Tamper Resistance

- **Definition:** The RoT must be resistant to physical, side-channel, and software-based attacks.
- **Why It Matters:** Protects against sophisticated adversaries attempting to extract keys or modify RoT.

6. Lifecycle Management

- **Definition:** The RoT must support secure updates and revocation mechanisms when vulnerabilities are discovered.
- **Why It Matters:** Allows the system to adapt to new threats without compromising security.

7. Non-Circumventable

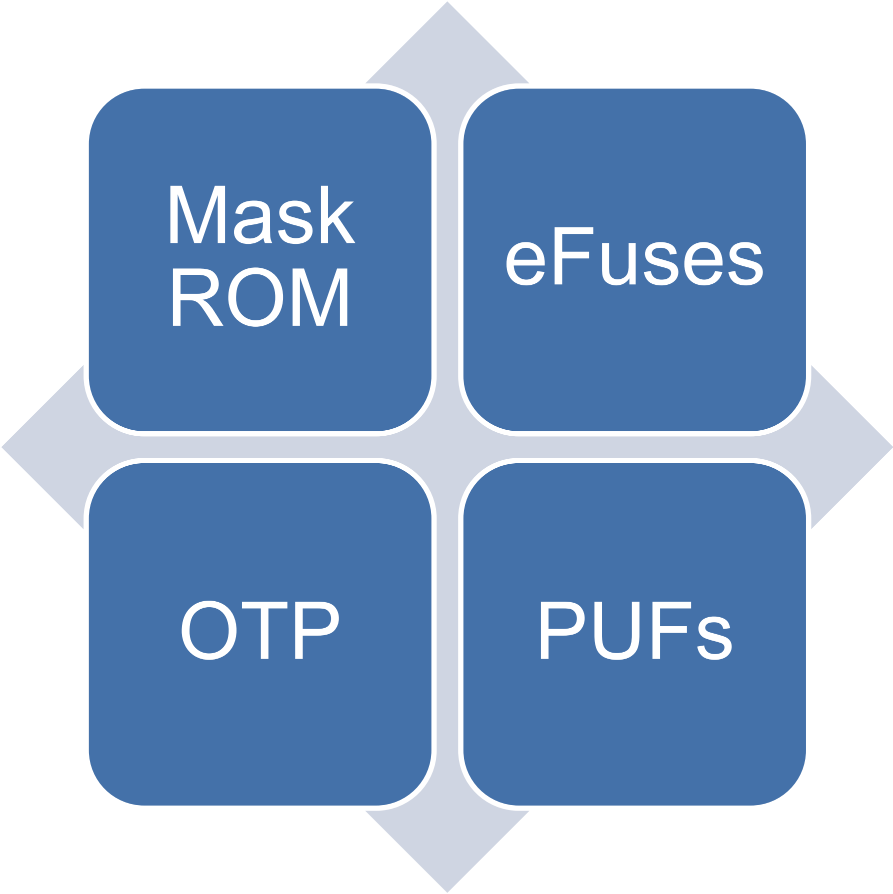
- **Definition:** The RoT must be designed so that it cannot be bypassed or overridden.
- **Why It Matters:** Guarantees that all processes depend on the RoT for establishing trust.

8. Anti-Rollback Protection

- **Definition:** The RoT must prevent the system from reverting to older, potentially vulnerable software versions.
- **Why It Matters:** Ensures system security is maintained even after updates.

SECURE BOOT OVERVIEW – TECHNOLOGIES USED

Often more than one of these technologies are chosen to establish the root of trust



SECURE BOOT OVERVIEW – ESTABLISHING A ROOT OF TRUST (1/3)

Roots of Trust

- Immutable code established ‘ground truth’
- Anchors boot integrity to hardware

1. Mask ROM

- **Description:** Read-only memory programmed into the chip at design time and used to store boot code or cryptographic functions.
- **Advantages:**
  - Highly tamper-resistant.
  - Ideal for storing trusted, unchangeable code.
- **Challenges:** Not updatable, so any vulnerabilities in the code are permanent. Requires a respin of the device if vulnerably discovered.

2. eFuses

- **Description:** Electrically programmable fuses used to store unique identifiers, cryptographic keys, or boot mode configurations.
- **Advantages:**
  - Can permanently store security-critical information.
  - Allows selective configuration of security features.
- **Challenges:** Irreversible programming and potential susceptibility to physical attacks.

RENESAS

SECURE BOOT OVERVIEW – ESTABLISHING A ROOT OF TRUST (2/3)

3. One-Time Programmable (OTP) Memory

- **Description:** Non-volatile storage medium that can only be written once.
- **Advantages:**
  - Permanently stores sensitive data.
  - Resistant to tampering since the data cannot be altered after initial programming.
  - Doesn’t require a device respin.
- **Challenges:** May requires secure programming at manufacturing.

4. Physical Unclonable Functions (PUFs)

- **Description:** Uses inherent physical variations in semiconductor manufacturing to generate unique, device-specific cryptographic keys.
- **Advantages:**
  - Keys are derived only when needed and not stored permanently, reducing attack surfaces.
  - Resistant to cloning and tampering.
- **Challenges:** Requires sophisticated design and calibration.

SECURE BOOT OVERVIEW – ESTABLISHING A ROOT OF TRUST (3/3)

5. Cryptographic Accelerators

- **Description:** Hardware modules designed to perform cryptographic operations (e.g., encryption, decryption, hashing, and signature verification) efficiently, enhancing the performance and security of secure boot processes.
- **Advantages:**
  - Offloads computationally intensive tasks from the CPU, improving boot time and energy efficiency.
  - Provides strong protection against software-based attacks by isolating cryptographic operations in dedicated hardware.
- **Challenges:** Increases hardware complexity and cost and limited to the algorithms chosen at design time.

SECURE BOOT CHALLENGES

RENESAS

SECURE BOOT CHALLENGES

Why is this difficult?

SECURE BOOT CHALLENGES – MANY HARDWARE VARIATIONS

Examples of different automotive hardware architectures. Sometime multiple types on same ECU.

MCUs with SHE

- No dedicated security core
- Limited boot options
- CMAC based boot

MCUs with HSM

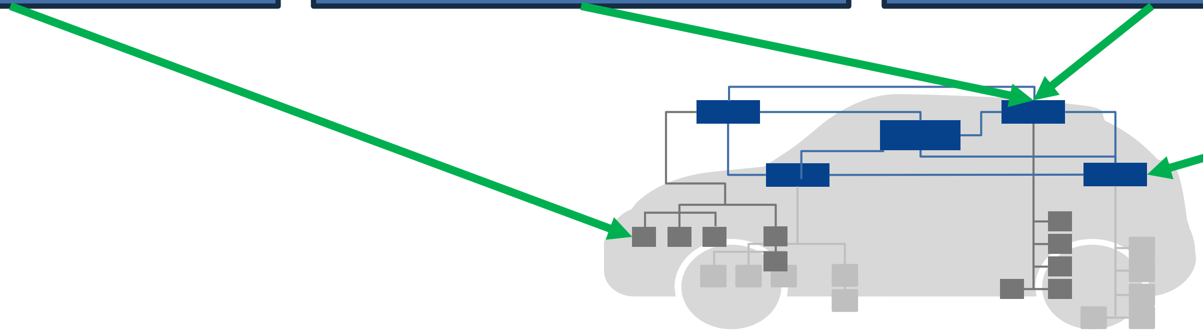
- Dedicated security core
- Flexible boot options
- Crypto agility

SoC w/o HSM

- Mask ROM based boot
- Limited flexibility for BL0
- Limited HW crypto support

SoCs with HSM

- Reduced mask ROM footprint
- Flexible boot options



Zonal architecture

SECURE BOOT CHALLENGES – BALANCING REQUIREMENTS

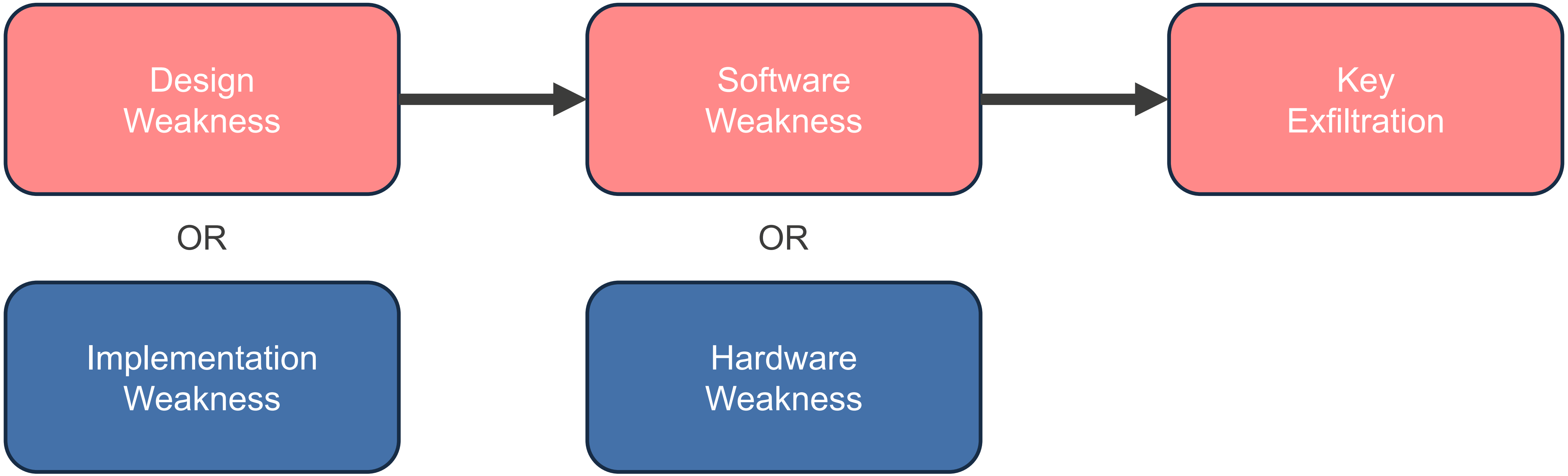
- Key Management
- Rollback Attacks
- Performance Requirements
- Sanctions
- Multi-core / Multi-device
- Policy

RENESAS

SECURE BOOT CHALLENGES – ATTACK SURFACE

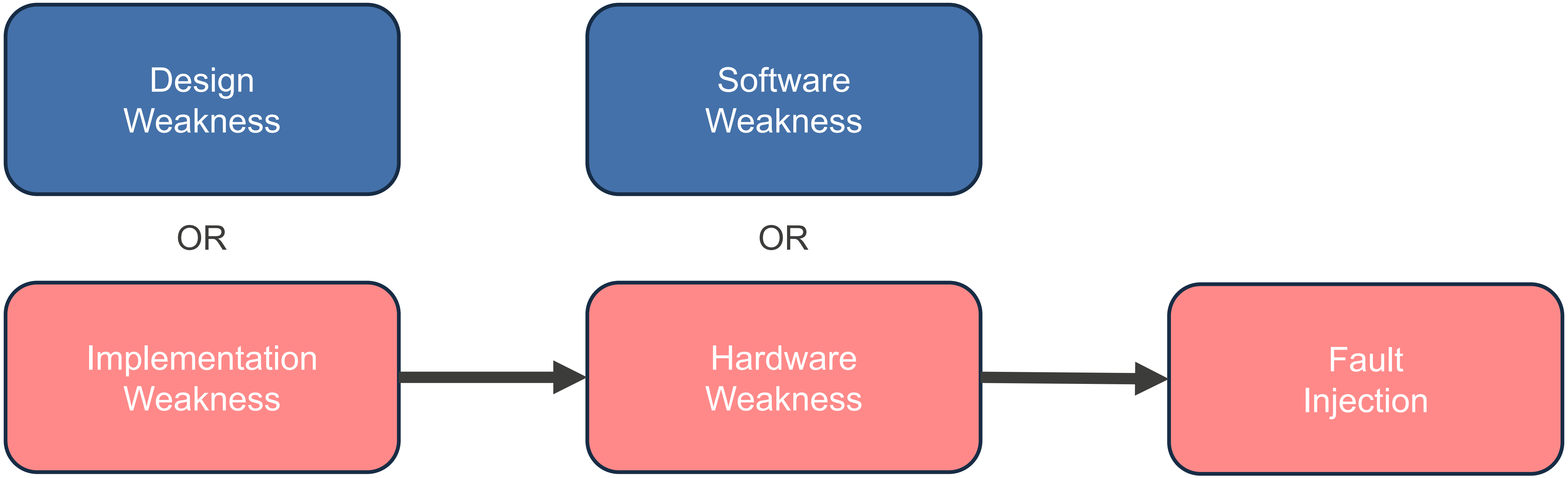
Design Weakness		Software Weakness
OR		OR
Implementation Weakness		Hardware Weakness

SECURE BOOT CHALLENGES – EXAMPLE ATTACKS ON BOOT



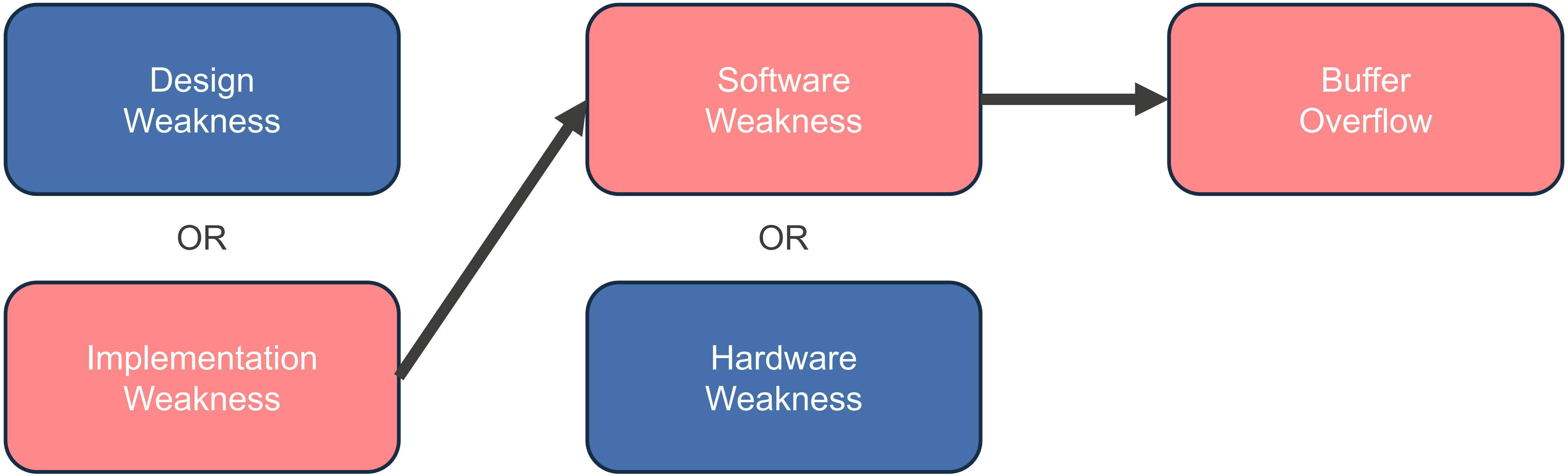
Timing side-channel allows recovery of decryption key

SECURE BOOT CHALLENGES – EXAMPLE ATTACKS ON BOOT



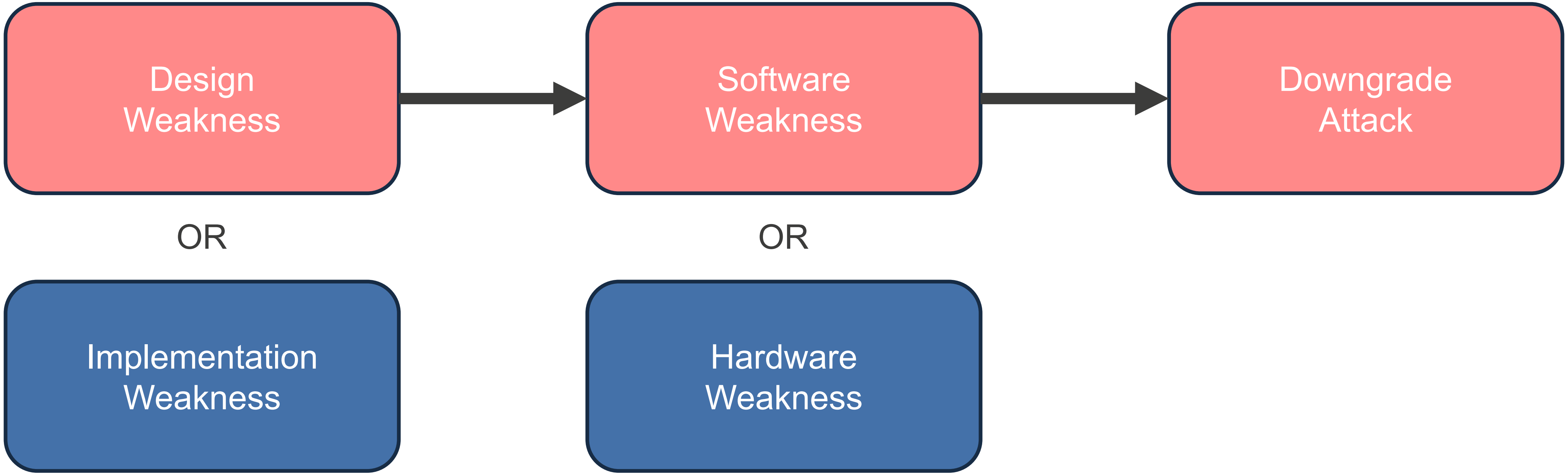
Fault injection allows bypass of secure boot

SECURE BOOT CHALLENGES – EXAMPLE ATTACKS ON BOOT



Buffer overflow allows bypass of secure boot

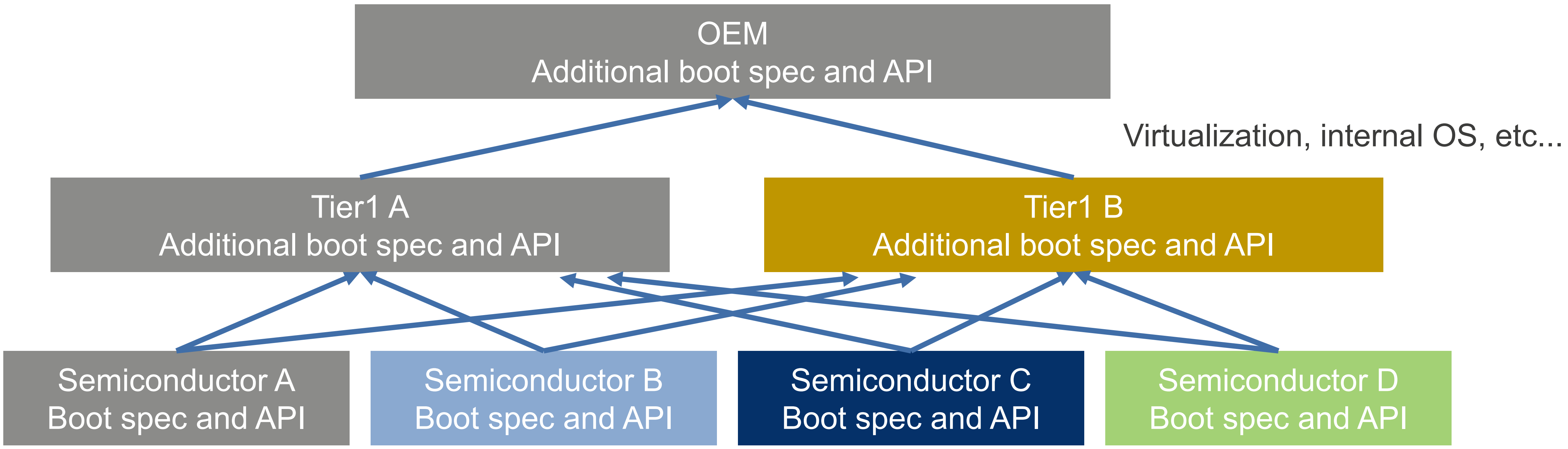
SECURE BOOT CHALLENGES – EXAMPLE ATTACKS ON BOOT



Design allows for downgrade from RSA to SHA1

WHAT COULD GLOBAL PLATFORM DO?

BUT FROM BIRDS EYE... (IN ONE VEHICLE)

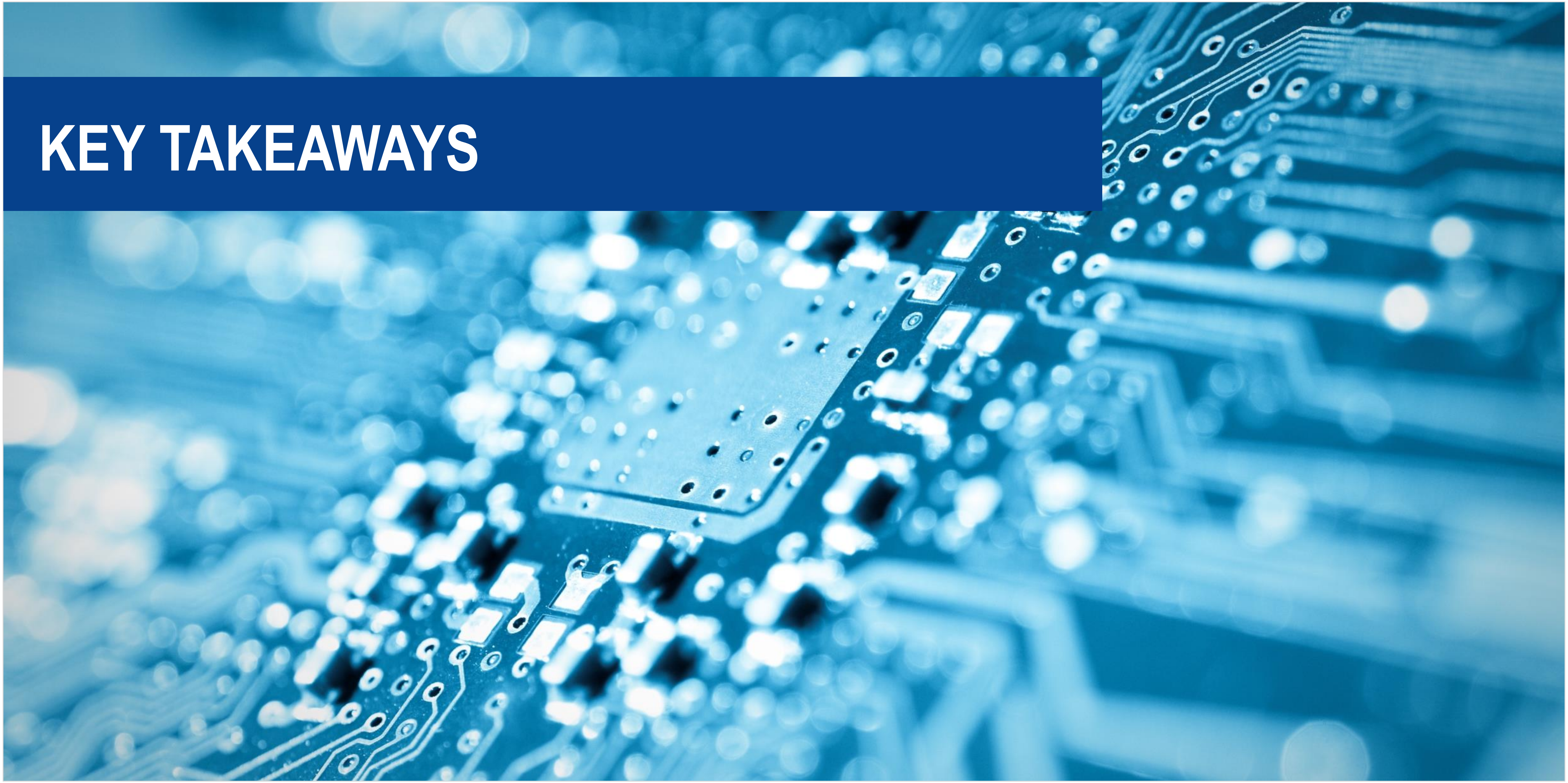


There is no rigorous regulation to specify
- Concrete (fixed) API
- Which cryptographic algorithm(s) to be used (more complex in PQC migration era)

HOW CAN GLOBAL PLATFORM ADDRESS THESE ISSUES?

- Designate secure boot as critical to Root of Trust function
- **Expand the Root of Trust requirements** to more explicitly define secure boot requirements
  - Cryptographic primitives
  - Immutable code requirements
  - Key management
- **Define the threat model and proposed mitigations**
  - Address threats like fault injection, replay attacks, tampering
  - Recommend mitigations or at least requirements around mitigations
- **Define boot policies and requirements** around reporting
- **Define performance and testing requirements** around secure boot

KEY TAKEAWAYS



KEY TAKEAWAYS

- **Importance of Secure Boot:**
  Detect and prevent unauthorized code execution during the boot process, establishing a device root of trust.
- **Root of Trust Properties:**
  Must include immutability, minimal scope, integrity/authenticity, confidentiality, tamper resistance, lifecycle management, non-circumventable, and anti-rollback protection.
- **Challenges and Solutions:**
  Address key security, rollback attacks, and performance requirements with mitigations like boot policies and secure update mechanisms.
- **GlobalPlatform's Role:**
  Could GlobalPlatform standardize secure boot by defining requirements, threat models, and mitigations, ensuring consistent implementation across platforms?


Thank You

VERSION HISTORY

- 2024-12-03: Initial release