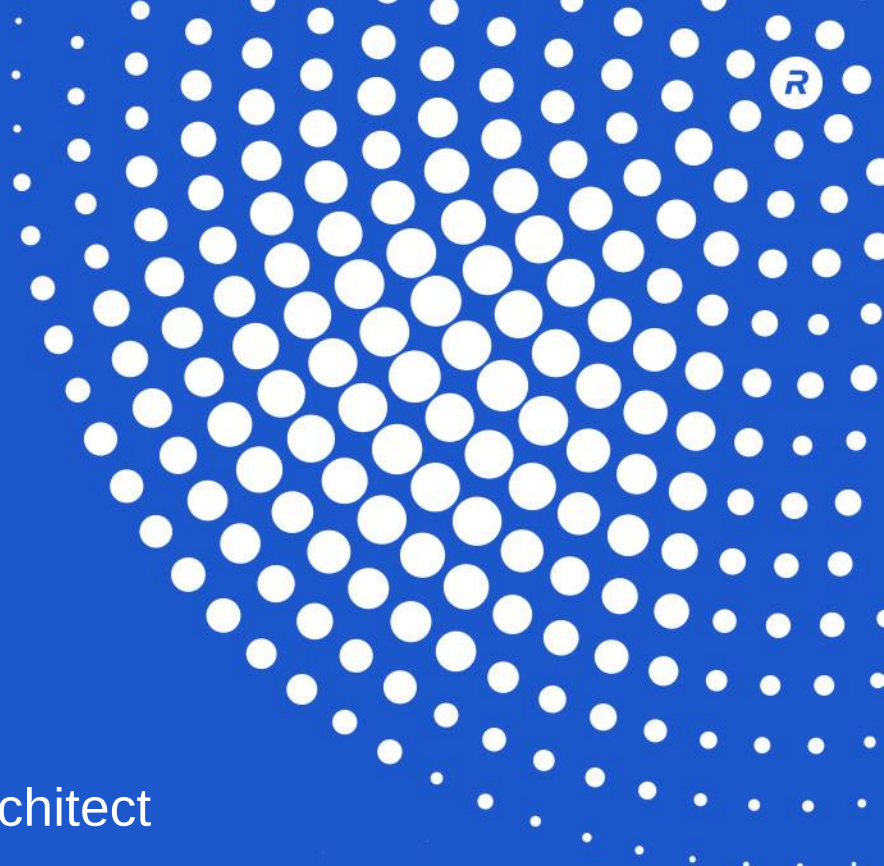
The Future of Security is in Open Silicon

Linux Security Summit 2018

Joel Wittenauer - Embedded Software Architect
Rambus Cryptography Research

August 28, 2018

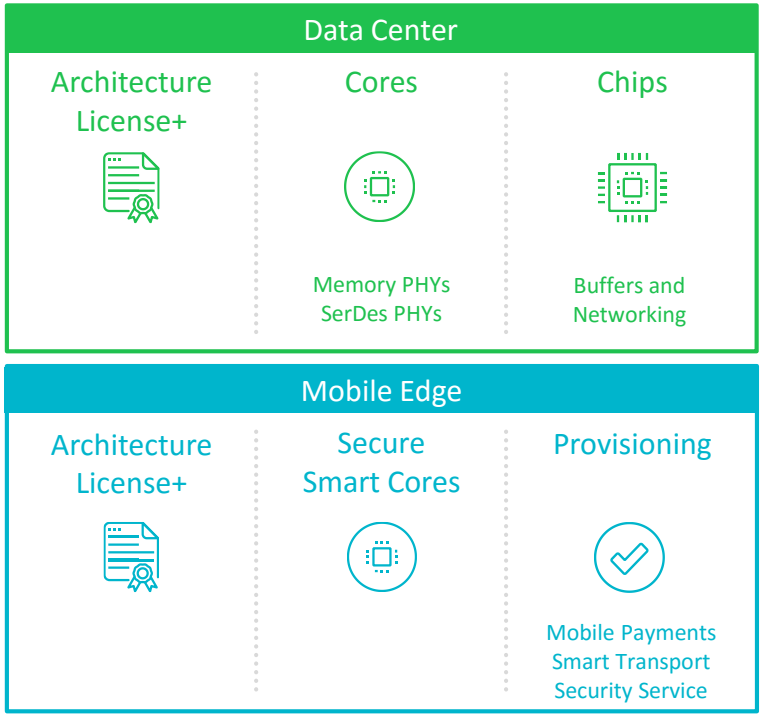


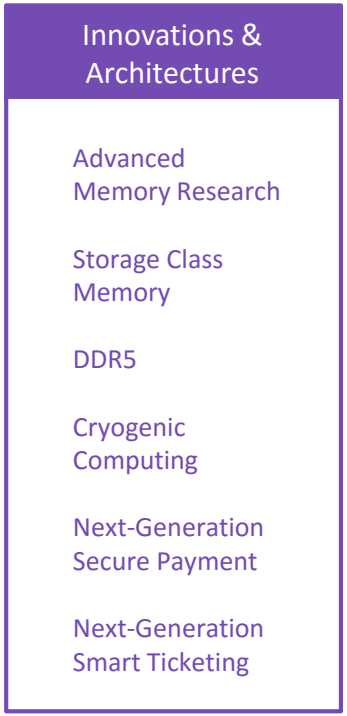
Rambus
Data · Faster · Safer

Agenda

- About Rambus Cryptography Research
- Spectre/Meltdown/Foreshadow
- Introduction to the CryptoManager Root-of-Trust (CMRT)
- CMRT Hardware
- CMRT Software
- Linux Applications

Rambus At-a-Glance





Corporate Facts

- NASDAQ: RMBS, Inc. in 1990; IPO 1997
- Headquartered in Sunnyvale, CA
- Operations throughout North America, Europe & Asia
- ~800 employees
- ~2,500 patents & applications

Markets Served

Data Center

- Acceleration + lower power
- System-level architectures
- Innovative products to support needs

Mobile Edge

- Secure endpoints = secure data
- Hardware root-of-trust
- Value and monetization for end-user services

Agenda

- About Rambus Cryptography Research
- Spectre/Meltdown/Foreshadow
- Introduction to the CryptoManager Root of Trust (CMRT)
- CMRT Hardware
- CMRT Software
- Linux Applications

Spectre/Meltdown/Foreshadow

- Spectre/Meltdown are vulnerabilities found in Intel, AMD and some ARM CPUs – reported in January 2018
- Foreshadow is a related vulnerability for Intel CPUs – reported August 2018
- Spectre research team included CRI founder Paul Kocher and Rambus security researcher Mike Hamburg
- All three vulnerabilities take advantage of CPU performance enhancements
  - E.g. speculative or out-of-order execution
- Documented in CVE-2017-5715, CVE-2017-5753, CVE-2017-5754, CVE-2018-3615, CVE-2018-3620 and CVE-2018-3646

SPECTRE

MELTDOWN

FORESHADOW

Spectre/Meltdown/Foreshadow

"…beyond short-term solutions such as patching, the semiconductor industry should seriously consider designing chips that run sensitive cryptographic functions in a physically separate secure core, silo-ed away from the CPU. This design approach will go a long way in helping to prevent vulnerabilities that can be exploited by Meltdown and Spectre*." – Mike Hamburg, Rambus security researcher

SPECTRE

MELTDOWN

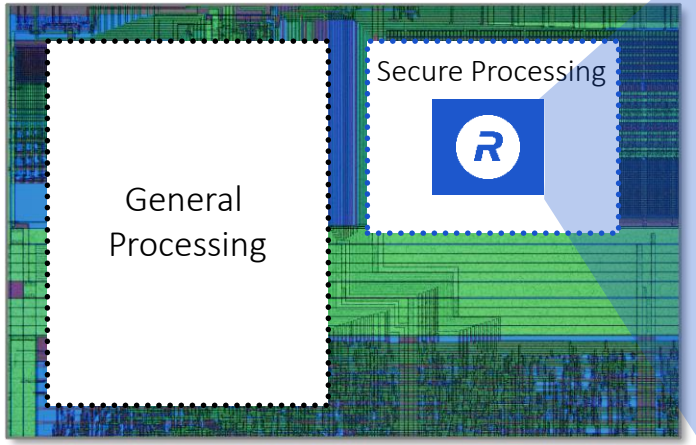
FORESHADOW

*Also Foreshadow

Agenda

- About Rambus Cryptography Research
- Spectre/Meltdown/Foreshadow
- Introduction to the CryptoManager Root of Trust (CMRT)
- CMRT Hardware
- CMRT Software
- Linux Applications

CryptoManager Root of Trust (CMRT)



CryptoManager Root of Trust

Custom RISC-V CPU

Secure Memory

Flexibility to run custom secure applications inside the CMRT boundary

Crypto Accelerators
(AES, SHA, others…)

Use Cases

CMRT can support a wide range of use cases, including:

- Secure data storage
- Secure key storage
- Device personalization
- Key and data provisioning
- Authentication
- Attestation
- User data privacy
- Secure boot
- Secure firmware update
- Secure communication
- Runtime integrity checking
- Cryptographic acceleration
- Secure protocol implementation
- Secure debug
- Feature/configuration/SKU management

Some brief CMRT terminology

Some brief CMRT terminology

- **Root**
  - Not a root user or certificate authority
  - For the CMRT, a root is an entity that is composed of an ID and permissions set for access to CMRT assets
  - The root defines the security context in which user applications execute

Some brief CMRT terminology

- **Root**
  - Not a root user or certificate authority
  - For the CMRT, a root is an entity that is composed of an ID and permissions set for access to CMRT assets
  - The root defines the security context in which user applications execute
- **Container**
  - Not related to Docker or other OS-level virtualization systems
  - For the CMRT, a container is a secure, user-privilege application that runs under the context of a root (defined above)

Some brief CMRT terminology

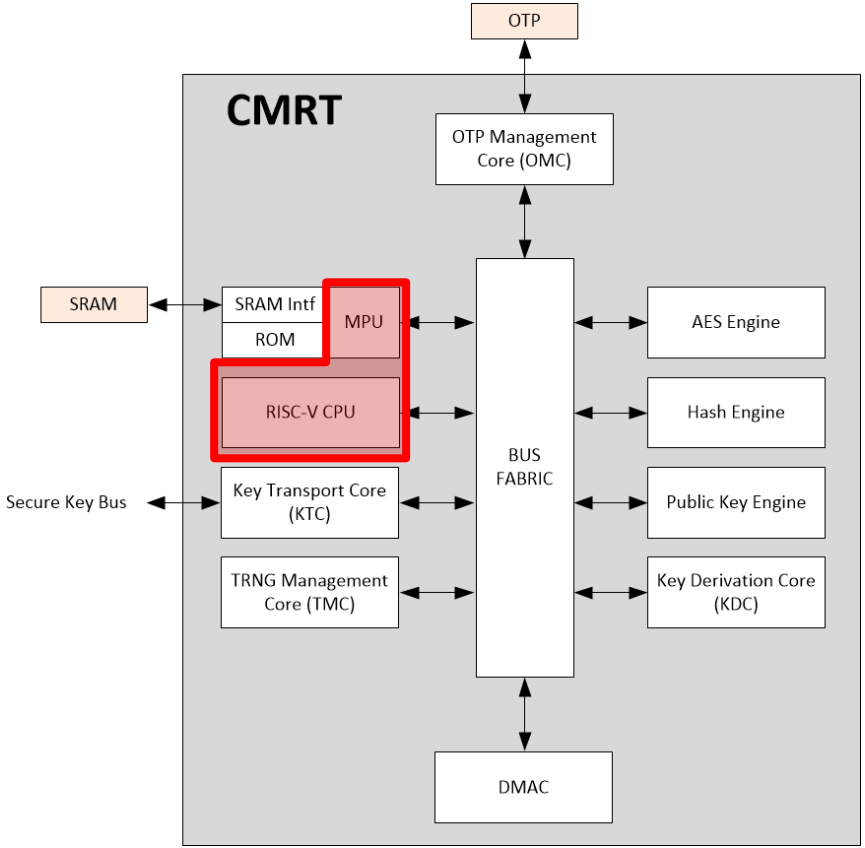
- **Root**
  - Not a root user or certificate authority
  - For the CMRT, a root is an entity that is composed of an ID and permissions set for access to CMRT assets
  - The root defines the security context in which user applications execute
- **Container**
  - Not related to Docker or other OS-level virtualization systems
  - For the CMRT, a container is a secure, user-privilege application that runs under the context of a root (defined above)
- Lesson: Don’t let hardware engineers name things

Agenda

- About Rambus Cryptography Research
- Spectre/Meltdown/Foreshadow
- Introduction to the CryptoManager Root of Trust (CMRT)
- CMRT Hardware
- CMRT Software
- Linux Applications

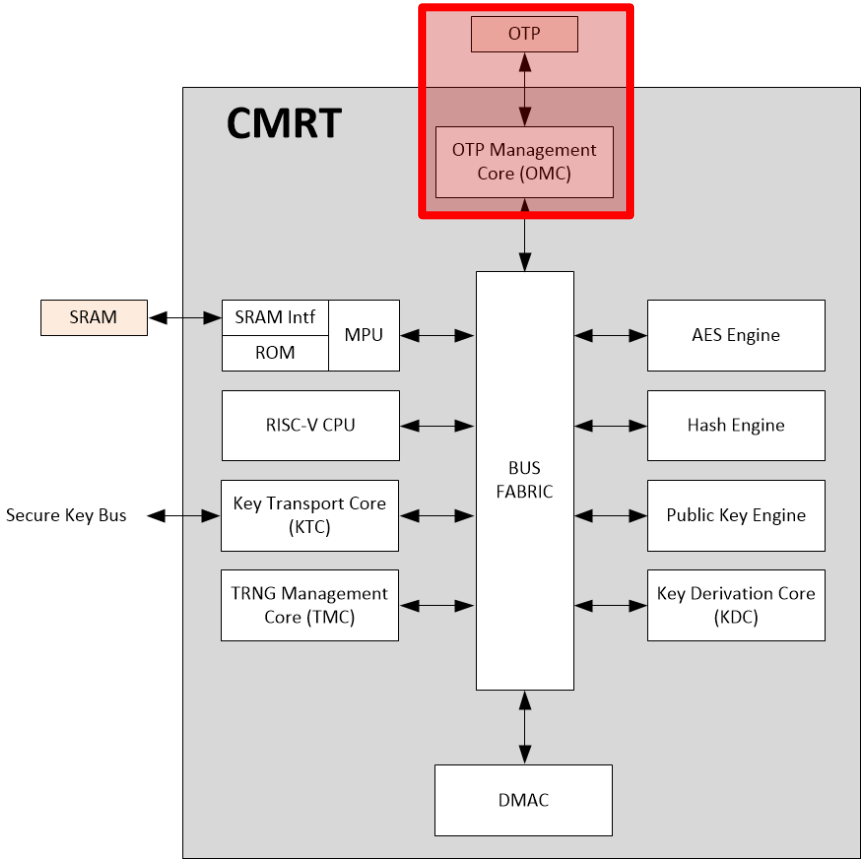
CPU/MPU

- CPU
  - A custom CPU designed by Rambus specifically for the CMRT
  - Based on the open-source RISC-V ISA and selected standard extensions
  - Supports three privilege levels – machine, supervisor, user
- Memory Protection Unit (MPU)
  - Sets regions of memory for access for one or more privilege levels (machine, supervisor or user) and access type (R,W,X)
  - MPU registers can be “locked” until the next CMRT reset



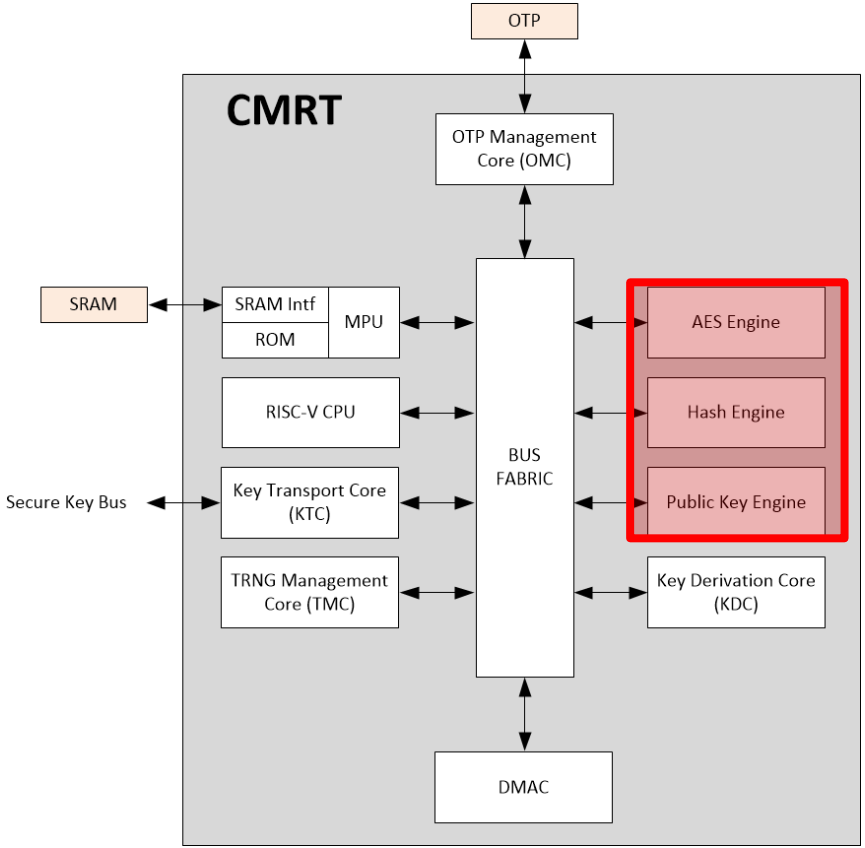
OTP

- CMRT uses One-Time Programmable (OTP) memory for non-volatile memory (NVM) storage
  - Writes of 0->1 are permanent
- OTP stores CMRT configuration:
  - Device ID
  - Lifecycle state
  - Device unique key
- A table of root IDs and permissions are stored in OTP
- General purpose NVM
  - Access to ranges of OTP addresses can be restricted by permissions



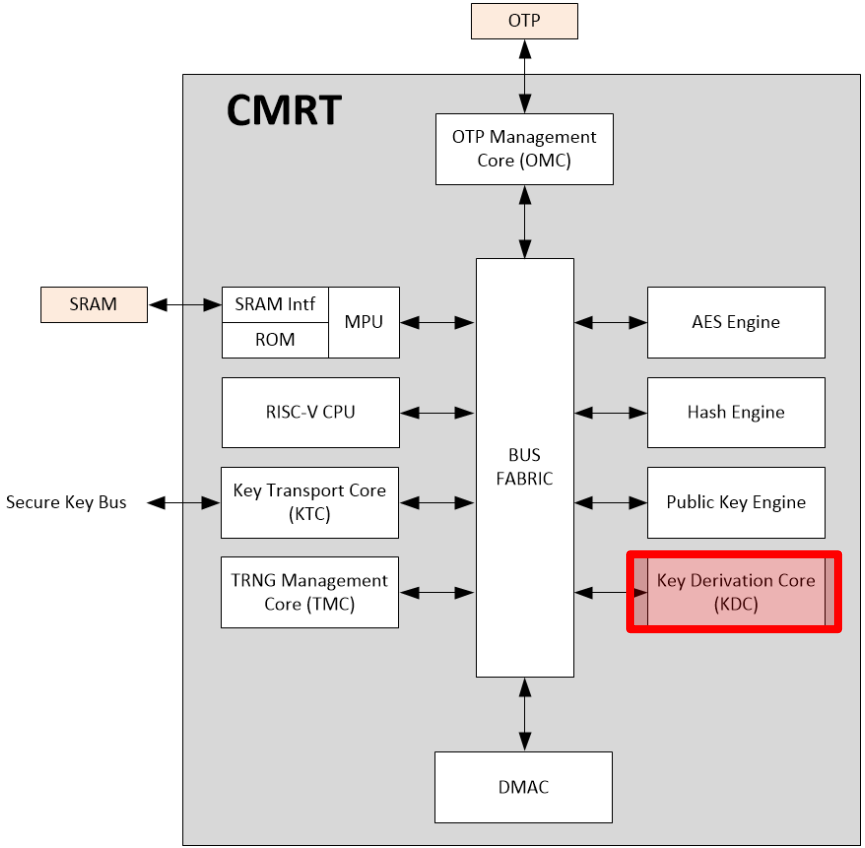
Crypto Engines

- Hardware AES core for data encryption/decryption
  - Multiple versions available, all with support for 128- and 256-bit keys:
  - Hardware support for different modes available, including ECB, CBC, CFB, CTR, GCM
- Hardware SHA-2 core
  - Supports both hashing and HMAC
  - SHA224/256/384/512
- Public key engine (PKE)
  - RSA/RSA-CRT 1, 2, 3 and 4K
  - ECDH and ECDSA with many curves:
    - NIST curves
    - Ed25519
    - Other curves available



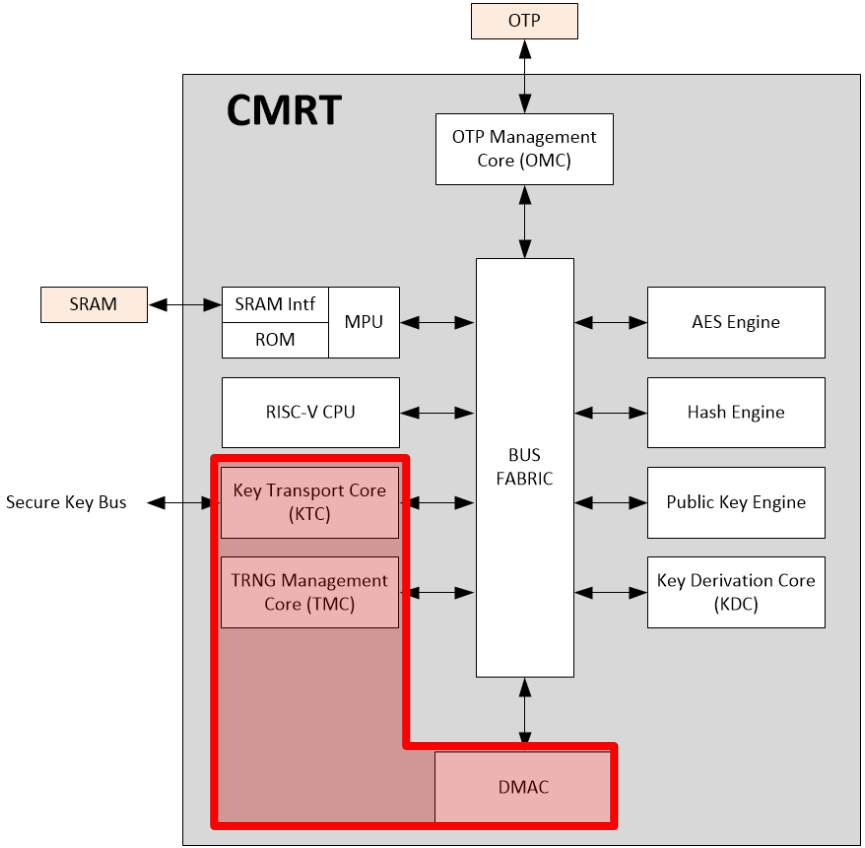
Key Derivation Core

- Responsible for deriving and managing keys
- Uses NIST-compliant key derivation algorithm to derive volatile keys from base keys
- Operates independent of CPU and can deliver keys to hardware cores without exposing them to CPU



Other important cores

- Key Transport Core (KTC) manages key interfaces outside the CMRT boundary
- NIST-compliant True Random Number Generator (TRNG)/Deterministic Random Bit Generator (DBRG)
- DMA Controller for fast movement of bulk data to/from SRAM, crypto cores, external memory, etc.

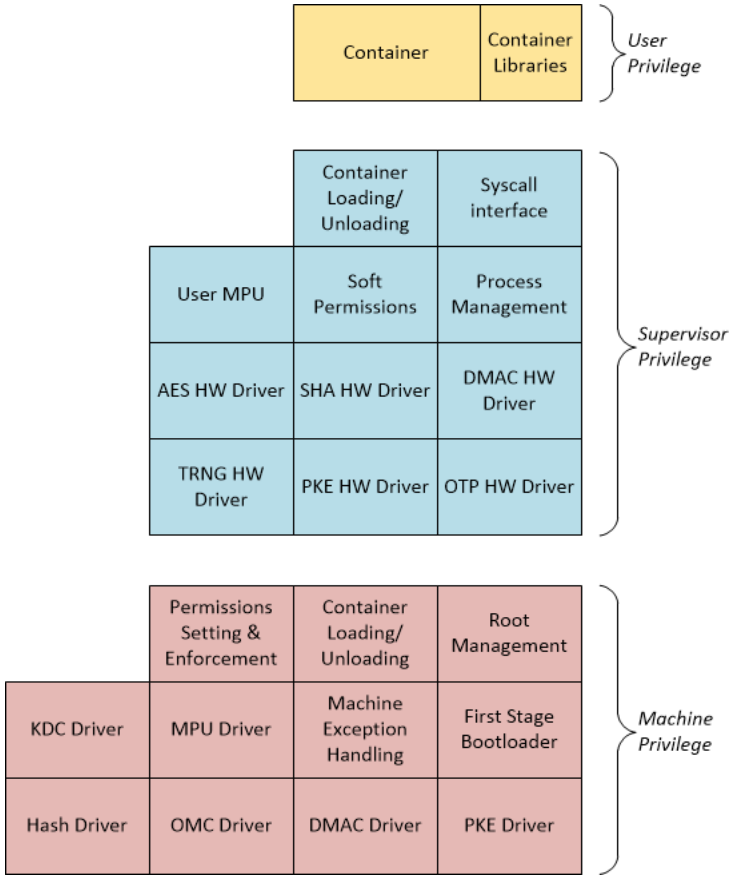


Agenda

- About Rambus Cryptography Research
- Spectre/Meltdown/Foreshadow
- Introduction to the CryptoManager Root-of-Trust (CMRT)
- CMRT Hardware
- CMRT Software
- Linux Applications

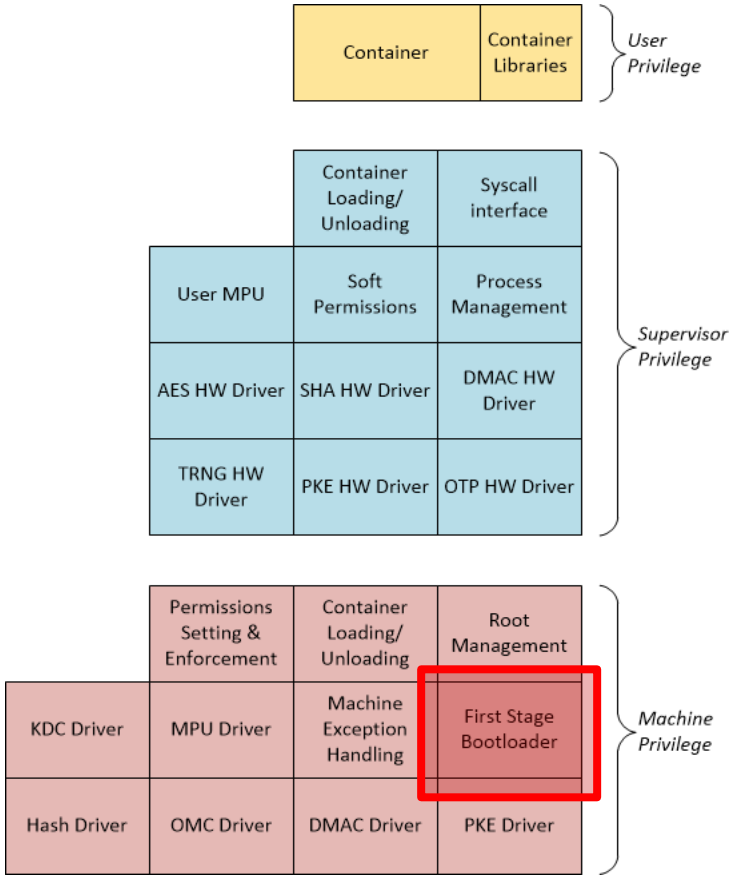
CMRT SW Architecture

- SW stack running on the CMRT CPU is divided into three privilege levels:
  - User: Containers
  - Supervisor: OS (Zephyr), HW drivers
  - Machine: Security-sensitive code/internal secure boot



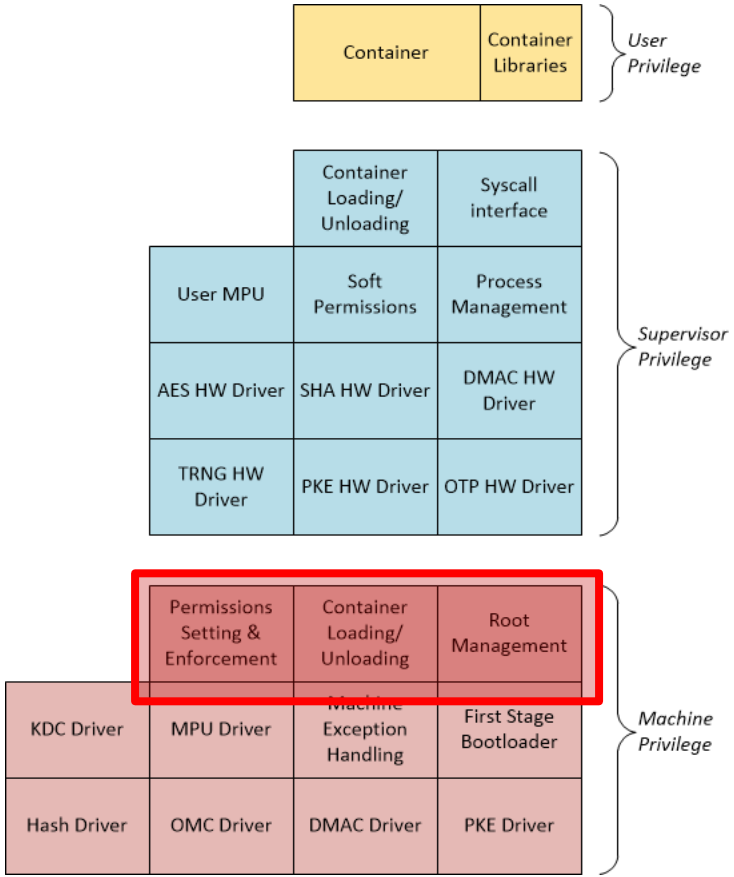
First-stage Bootloader

- Located in ROM
- Begins the secure boot of the CMRT
- A chain of trust is built from ROM, through images in OTP and flash
- Subset of device drivers included with bootloader



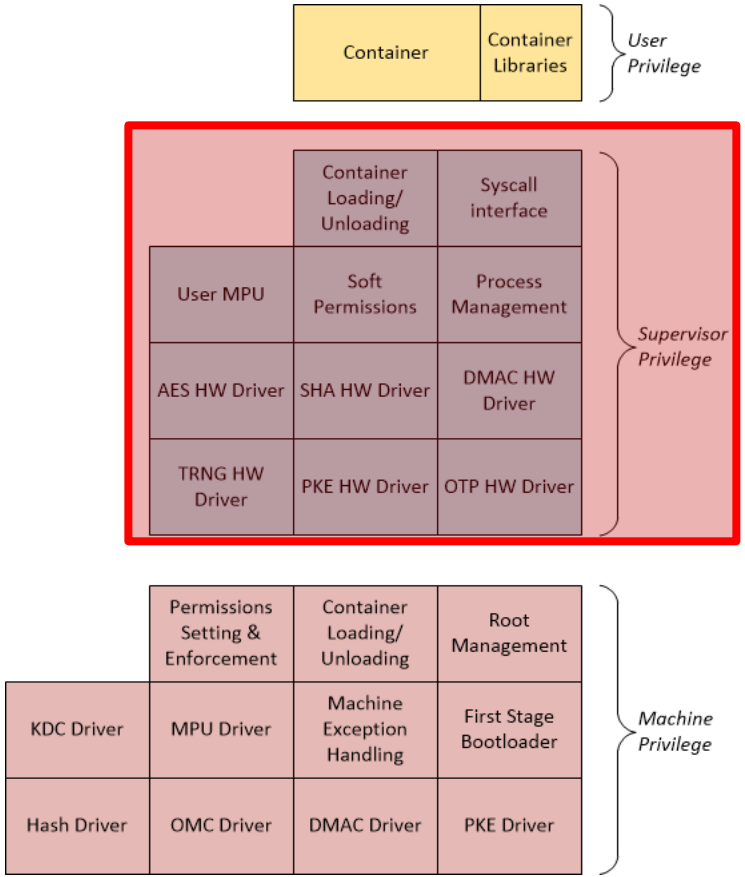
Security Monitor

- The most security-sensitive code is stored in trusted flash image
  - Securely loaded during CMRT boot process
- When a user container is loaded by the CMRT, the security monitor:
  - Verifies root “ownership” of the container via root table in OTP
  - Verifies the digital signature of the container code
  - Verifies the permissions requested by container are less than or equal to root
  - Applies hardware permissions to respective cores
- Handles root management



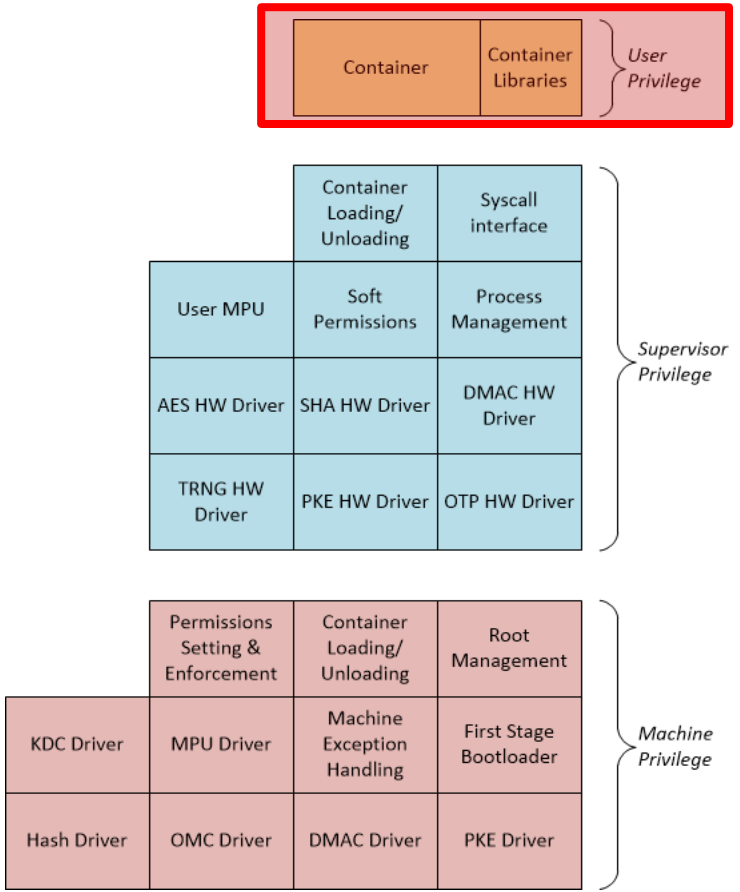
Supervisor OS/Kernel

- Modified version of the Zephyr OS
  - Supports application loading
  - Kernel/user memory separation
  - *nix-like device driver interface
    - open/close/ioctl…
  - All Rambus modifications are controlled by Kconfig
- Provides container's access to HW cores
- First line of permissions enforcement
  - Including some software-only permissions



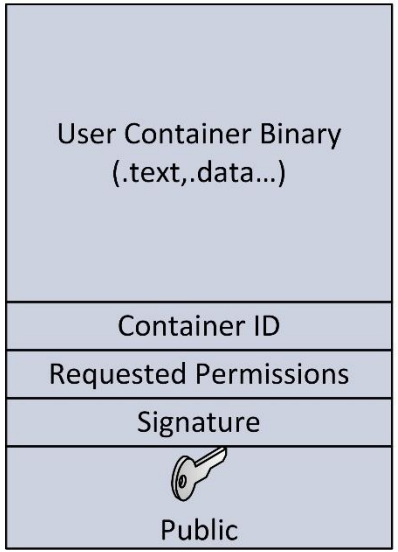
Containers

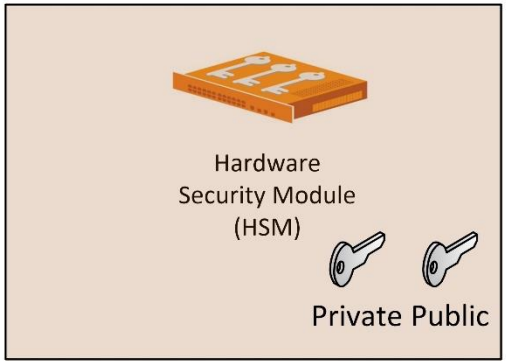
- Containers are customer-developed applications
- Each container is signed with a private key associated with a specific root
- Containers have associated permissions that control access to keys and hardware resources
- Permissions are also limited in HW to those available to the root associated with container
- Container libraries include C-runtime and a Global Platform TEE-compliant crypto library

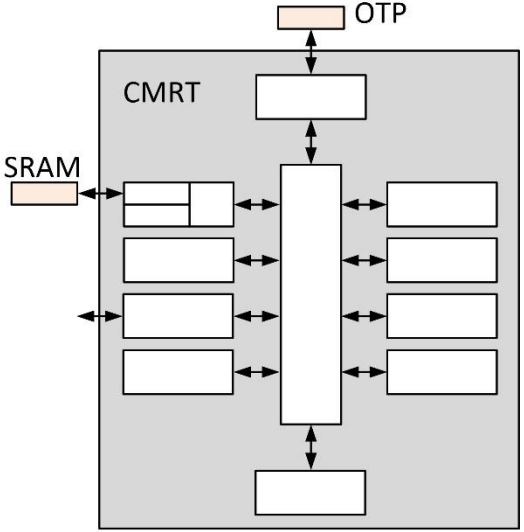


Building and executing containers

Root Permissions

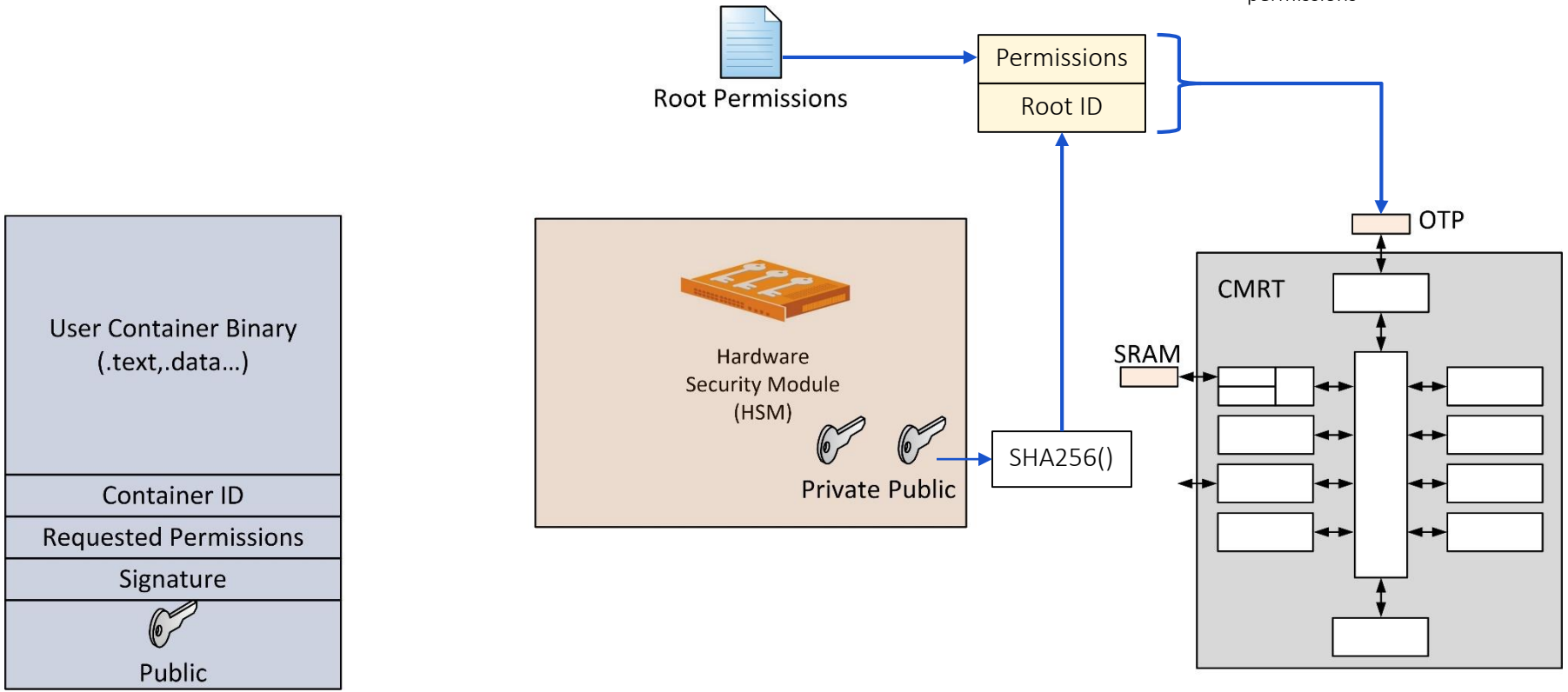






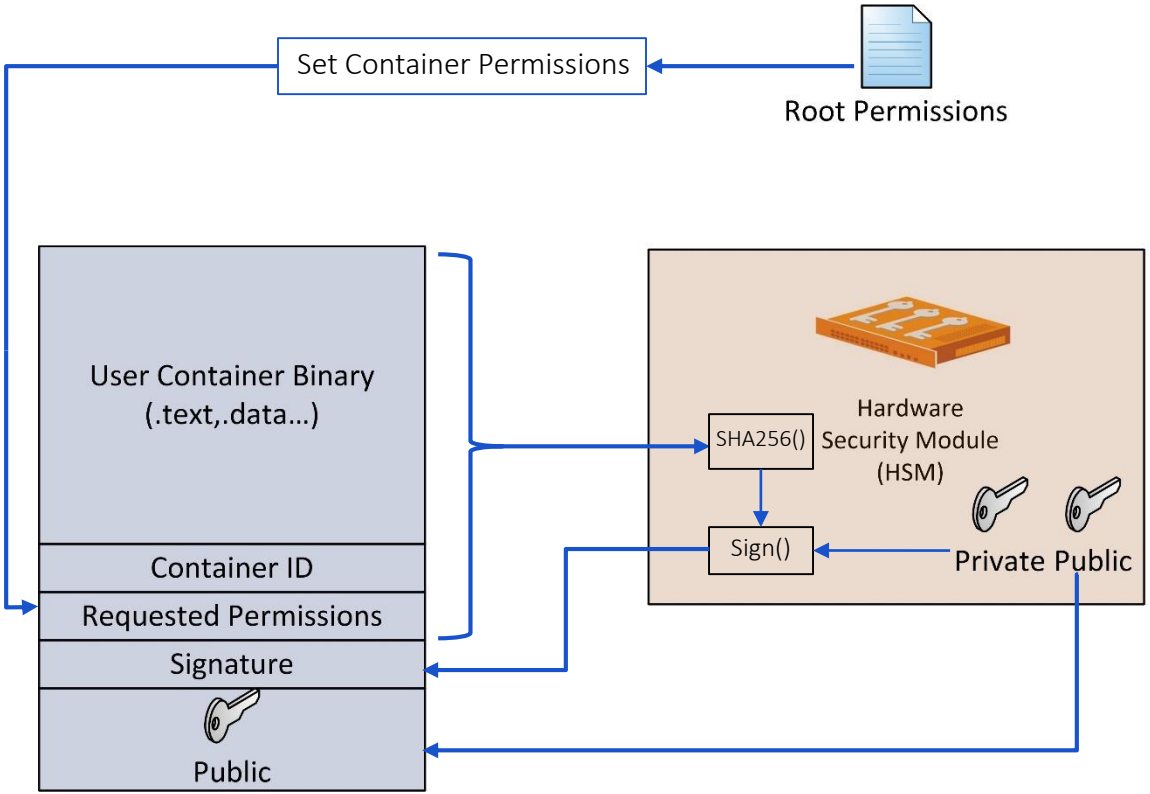
Building and executing containers

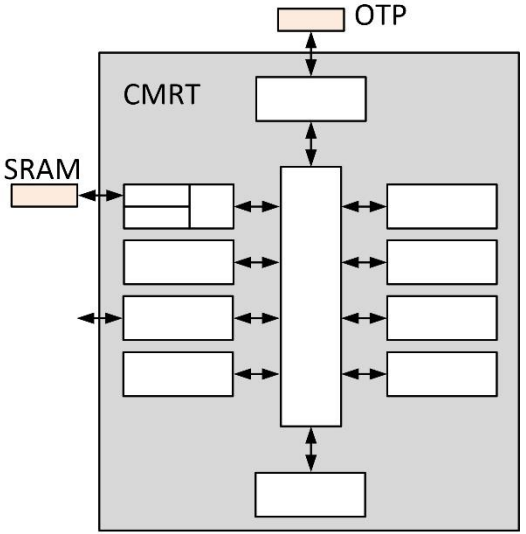
- Roots are provisioned during device manufacturing
- Can also be done later by containers with sufficient permissions



Building and executing containers

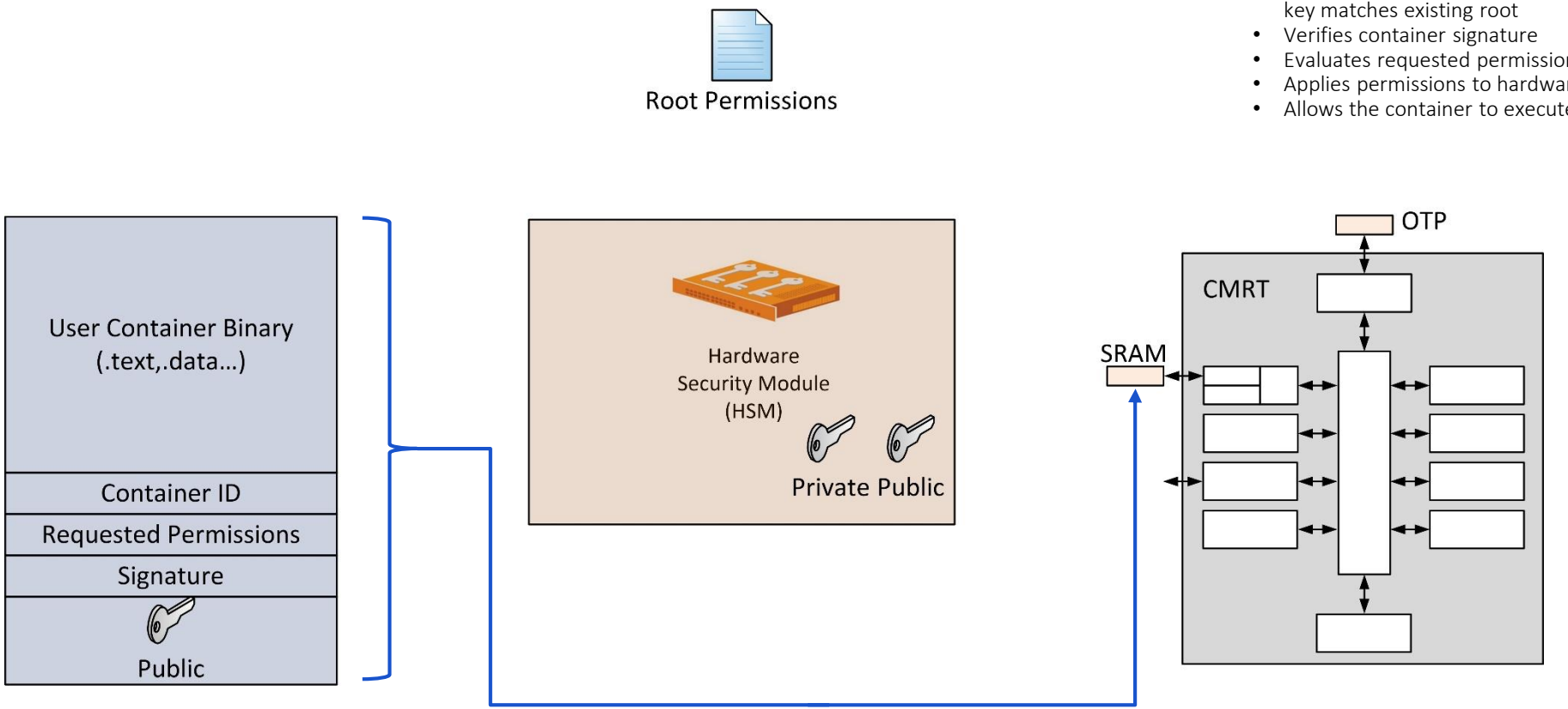
- Container permissions must be a subset (less than or equal) of the root's permissions
- After building the container attach footer with
  - Container ID
  - Requested permissions
- Sign the binary + Footer
- Attach the signature and public key to the footer





Building and executing containers

- Copy the container binary+footer to the CMRT's SRAM
- The CMRT:
  - Determines if container's public key matches existing root
  - Verifies container signature
  - Evaluates requested permissions
  - Applies permissions to hardware
  - Allows the container to execute



Why Zephyr?

- Existing RISC-V port
- Zephyr is a Linux Foundation project
  - Includes a Formal Governing Board, Technical Steering Committee and Security Committee
- Great ecosystem
  - Large, active number of contributors
  - Proper QA and CI in place
  - Community guidelines
  - Contribution reviews
- All CMRT software is built using the flexible Zephyr build system
  - Bootloaders, security monitor, OS/kernel and containers

Agenda

- About Rambus Cryptography Research
- Spectre/Meltdown/Foreshadow
- Introduction to the CryptoManager Root-of-Trust (CMRT)
- CMRT Hardware
- CMRT Software
- Linux Applications

CMRT Linux SDK

- Rambus offers a CMRT Linux Software Development Kit (SDK)
- The SDK provides:
  - Full container development environment
  - Out-of-tree reference kernel module implementations
  - RISC-V GCC 7 suite
  - CMRT QEMU emulator for fast container development
    - Enables development prior to the hardware availability
    - Easier to use/debug than FPGAs
    - Easier to scale CI
  - Reference implementation available on the Xilinx Zynq-7000 evaluation board

CMRT Device Driver

- The SDK's reference Linux device driver:
  - Loads/unloads trusted containers
  - Provides an interface for userspace applications to reach the CMRT
- Applications are limited only by one's imagination

CMRT Device Driver

- The SDK's reference Linux device driver:
  - Loads/unloads trusted containers
  - Provides an interface for userspace applications to reach the CMRT
- Applications are limited only by one's imagination
  - …and available CMRT SRAM
  - Note that there are mechanisms available to "chain" CMRT containers together to get around SRAM limitations

Other possible applications

- Linux crypto engine module
  - Similar to examples in drivers/crypto/*/
  - Requires a general purpose crypto container with a compliant interface
- Trusted Platform Module Emulation
  - Similar to examples in drivers/char/tpm/
  - Reduces need for extra part on PCB
  - Requires a container that exposes the TPM interface

Questions?

Thank you!

Joel Wittenauer
joel.wittenauer@cryptography.com
Rambus Cryptography Research