BedRock Systems Inc

Minimize your TCB using a Microkernel-Based System

Udo Steinberg

Agenda

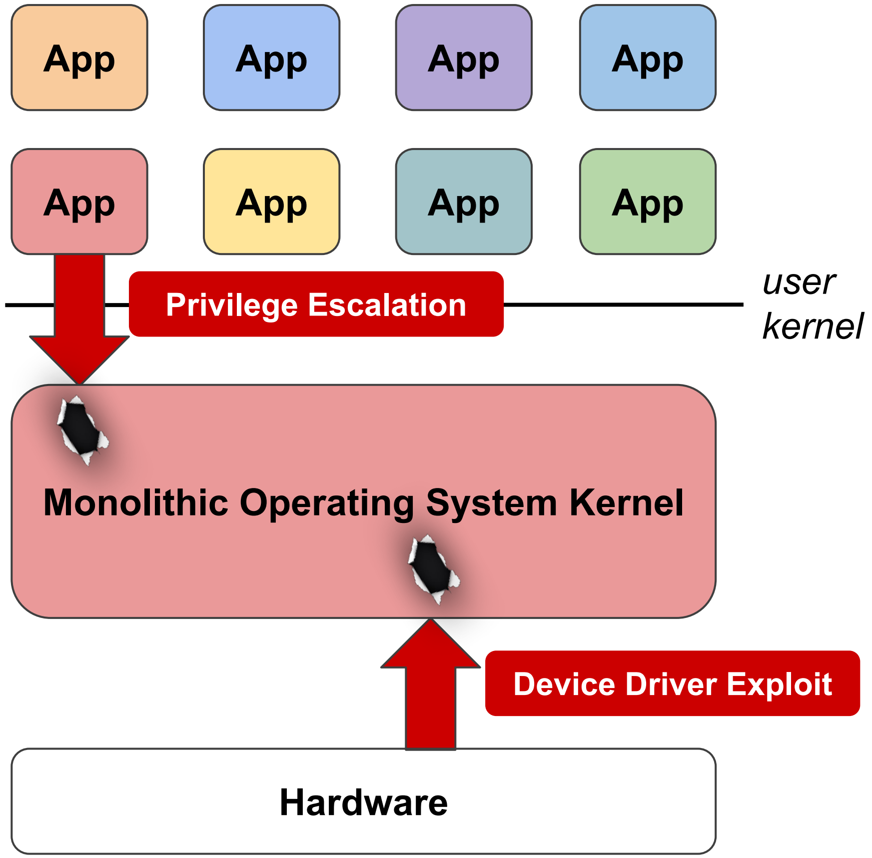
- The Fundamental Flaw in Today's Security Model
- Building a Trustworthy Trusted Computing Base
  - Microkernel / Microhypervisor
  - Capability-Based Access Control
  - Formal Verification
  - Active Security
- Advanced x86 Security Technologies
- Q & A

Trusted Computing Base

- “A **small** amount of software and hardware that security depends on and that we can distinguish from a much larger amount that can misbehave without affecting security” (B. Lampson)
- From a security perspective it is desirable to
  - Minimize the Trusted Computing Base (TCB)
  - Implement Fine-Grain Functional Disaggregation (Modularity)
  - Enforce the Principle of Least Authority (POLA)
- Size of the TCB is application-specific

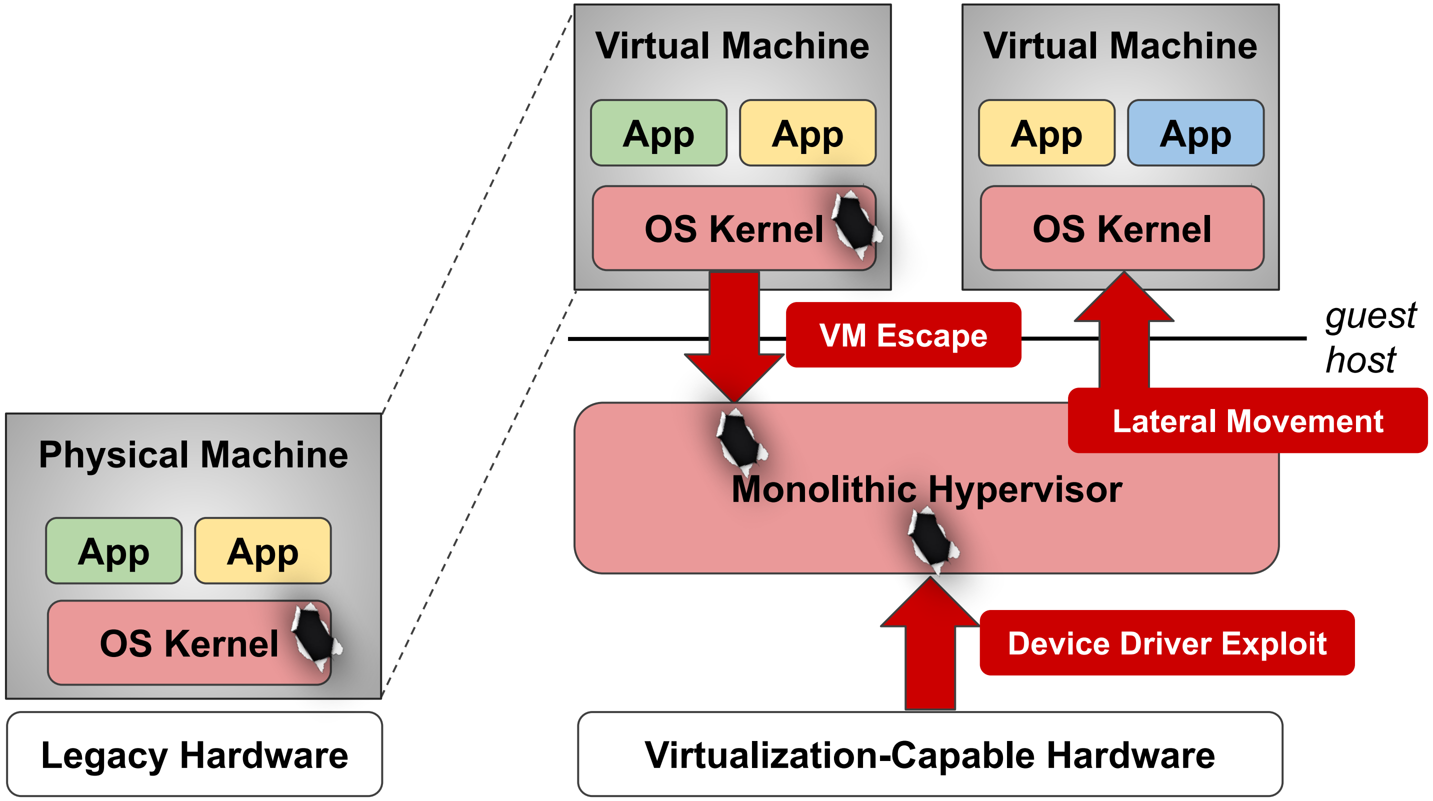
The Fundamental Flaw in Today's Security Model

- Significant parts of the code base are trusted, **but not trustworthy**
  - Millions of SLOC in modern kernels, ⅔ of it in device drivers (Linux 6.8: ~25 million)
- Huge attack surface for code running with **highest execution privileges**
  - Security controls can be silently disarmed because they run at the same privilege level that they are trying to protect



Virtualization / Operating System Encapsulation

- Using virtualization replaces physical with logical isolation
- Hypervisor layer increases the TCB size further
- Existing security problems move one layer down
- What have we gained?



Summary: Castles Built on a Foundation of Sand

- Complex systems software with exploitable security vulnerabilities
- **Defenders operate at the same privilege level as attackers**
- Contemporary security software can be subverted by kernel-mode malware
- **Traditional security model is failing** against advanced attacks

BedRock Systems Inc

BedRock Systems

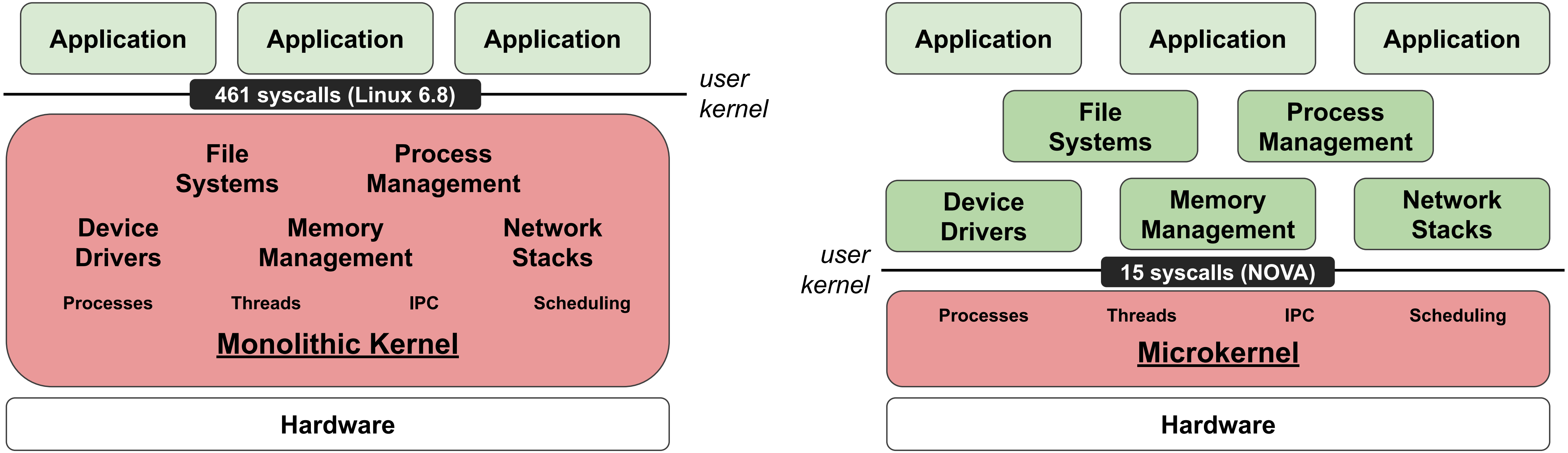
Next-Generation Workload & Runtime Security

BedRock Systems

- **Silicon Valley Based, Venture Capital Funded Startup**
  - Highly distributed: HQ in San Francisco, offices in Boston, Germany, Bangalore, …
- **Operating Systems Experts**
  - Building a very small and trustworthy TCB (around the NOVA Microhypervisor)
- **Formal Methods Experts**
  - Proving mathematically that the BedRock TCB conforms to its specifications
- **Security Experts**
  - Using the BedRock TCB to introspect and harden VMs and container runtimes

Making the TCB Trustworthy

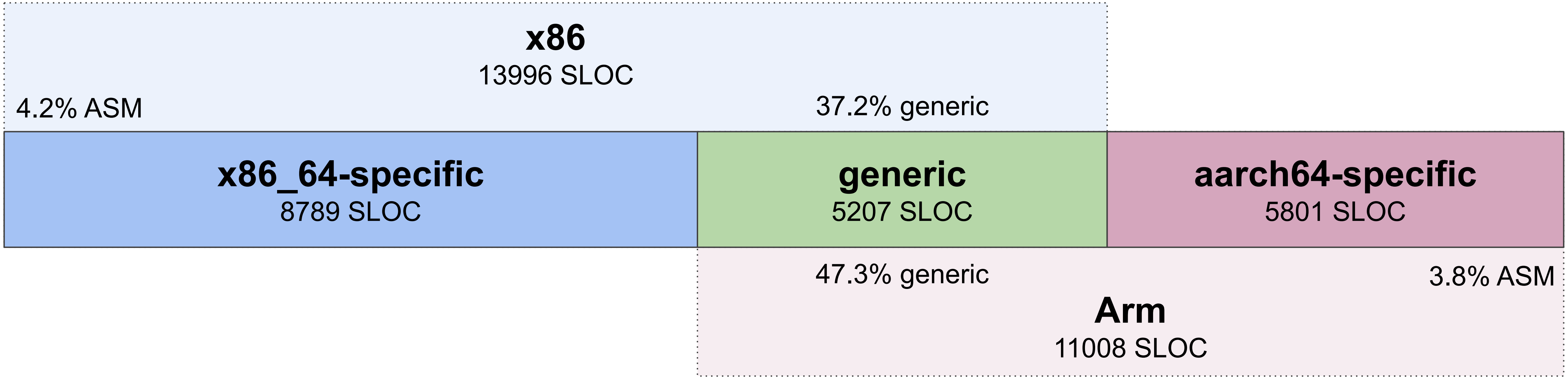
- Using a Microkernel instead of a Monolithic Kernel
  - Reduces the TCB size by **more than 2 orders of magnitude**
  - Enforces modularity and well-defined interfaces ⇒ Formal Verification becomes feasible



Microkernel Construction Principle

- “A concept is tolerated inside the microkernel only if moving it outside the kernel, i.e. permitting competing implementations, would prevent the implementation of the system’s required functionality” (J. Liedtke)
- Design Goals
  - Make the microkernel as small and fast as possible
  - Provide only mechanisms (but no policies) in the microkernel
  - Implement most system functionality in deprivileged user-mode components
  - Enforce the principle of least authority among all user-mode components (zero trust)

NOVA: Portable Unified Code Base (x86/Arm)



NOVA x86 ELF Binary

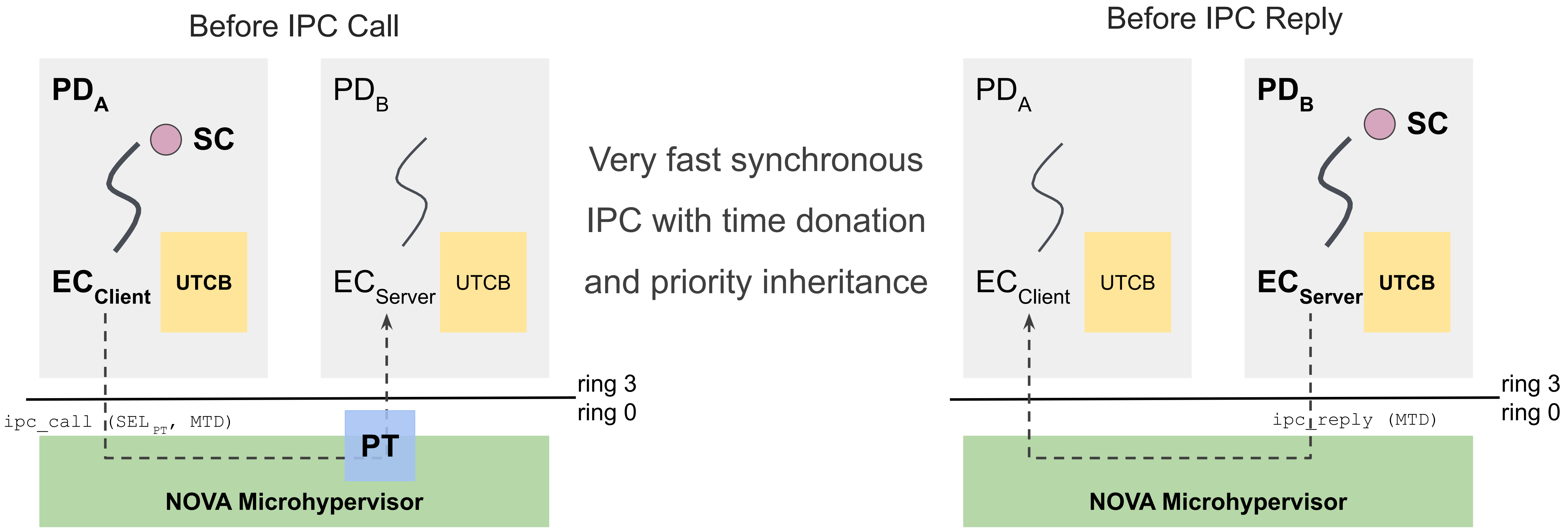
- 86377 Bytes Code
- 2520 Bytes Data

NOVA Arm ELF Binary

- 77896 Bytes Code
- 328 Bytes Data

SLOC based on **release-24.17.0**, binary sizes based on **gcc-13.2.0** build. Other versions will produce different numbers.

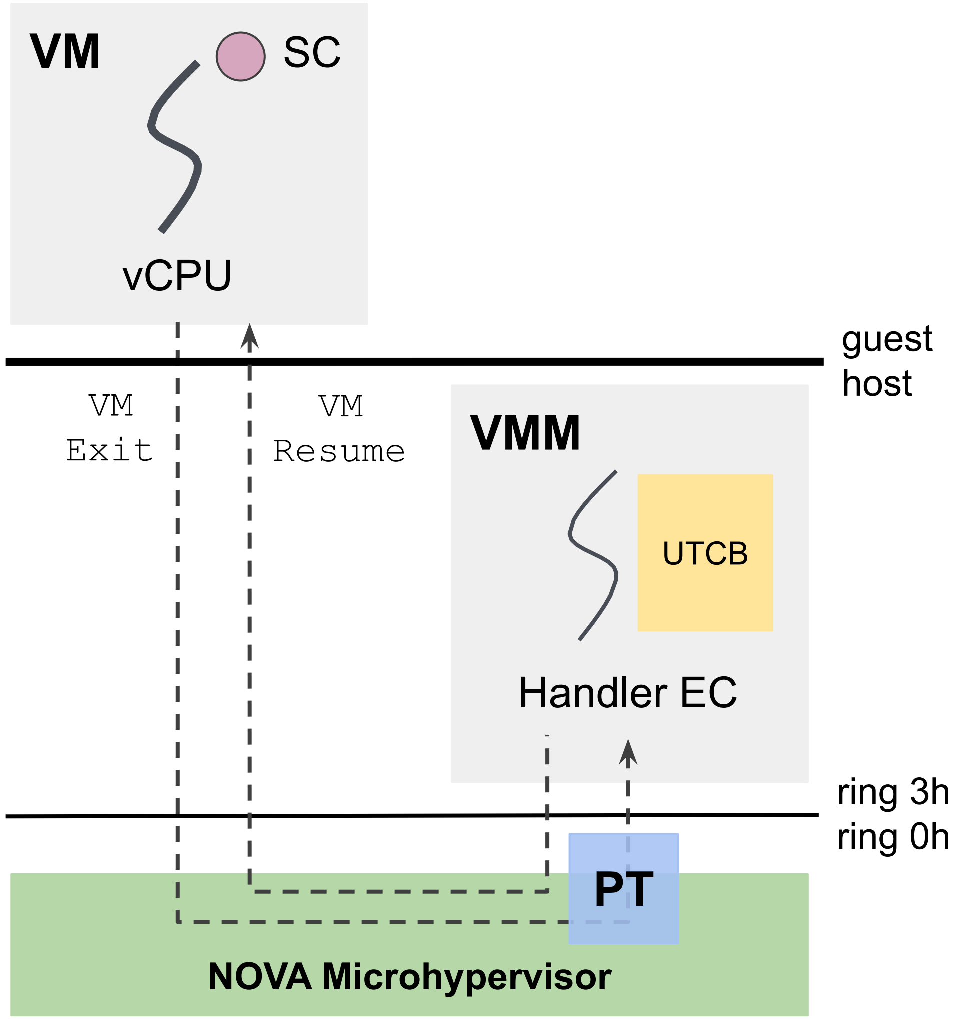
Microkernel Building Blocks



Very fast synchronous IPC with time donation and priority inheritance

- Protection Domains, Execution/Scheduling Contexts, Portals, Semaphores

From Microkernel to Microhypervisor



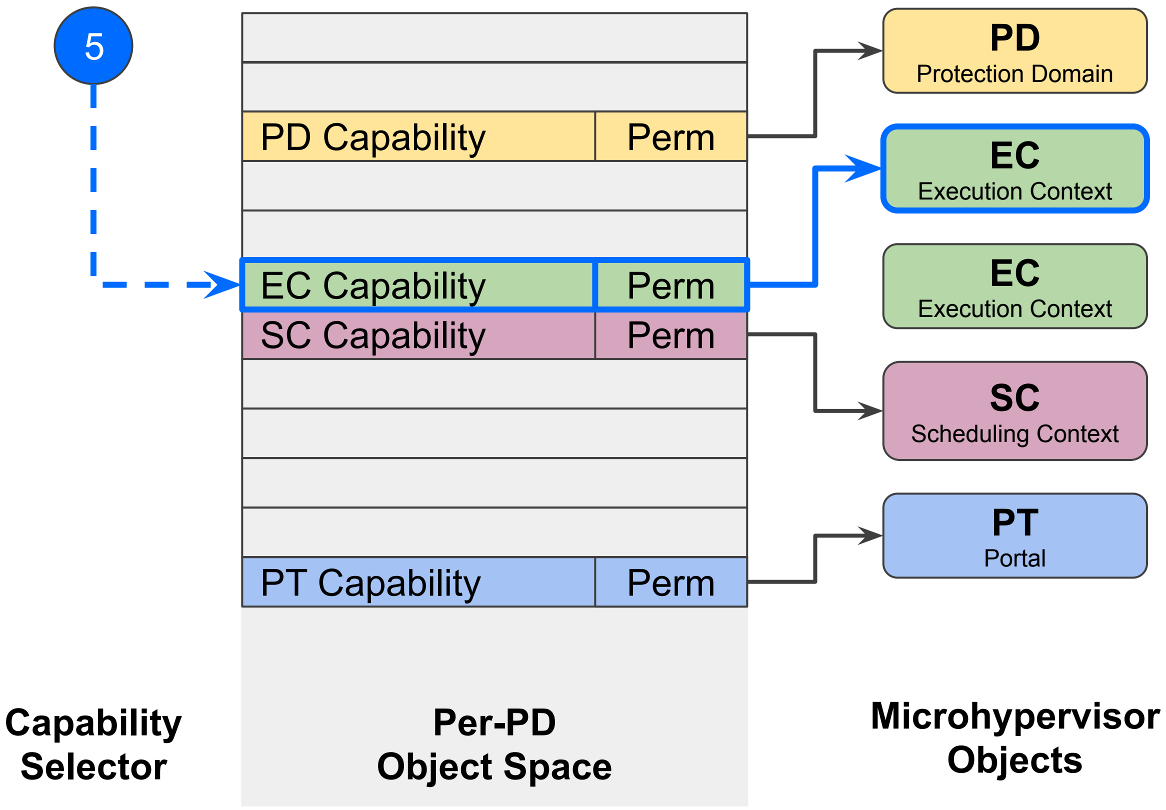
- Microkernel interface is not POSIX-compliant
- Reuse of legacy operating systems via VMs
- Deprivileged Virtual-Machine Monitor (VMM)
  - VM exits are forwarded to the user-mode VMM for handling – instruction and device emulation
  - Per-event portal defines subset of architectural state that NOVA transmits to the VMM's UTCB
  - VMM responds with updated state in its UTCB and optionally an event to inject

NOVA Microhypervisor Functionality

- Enumerates platform resources using UEFI/ACPI
- Manages security-critical functions of the platform
  - CPU, FPU, VMCS, MMU, SMMU (IOMMU), Interrupt Controllers (LAPIC, IOAPIC, GIC)
- Enforces spatial and temporal isolation between host components and VMs
  - Each component runs in its own address space
  - Preemptive fixed-priority round-robin core-local scheduler
- Provides very fast core-local communication via IPC

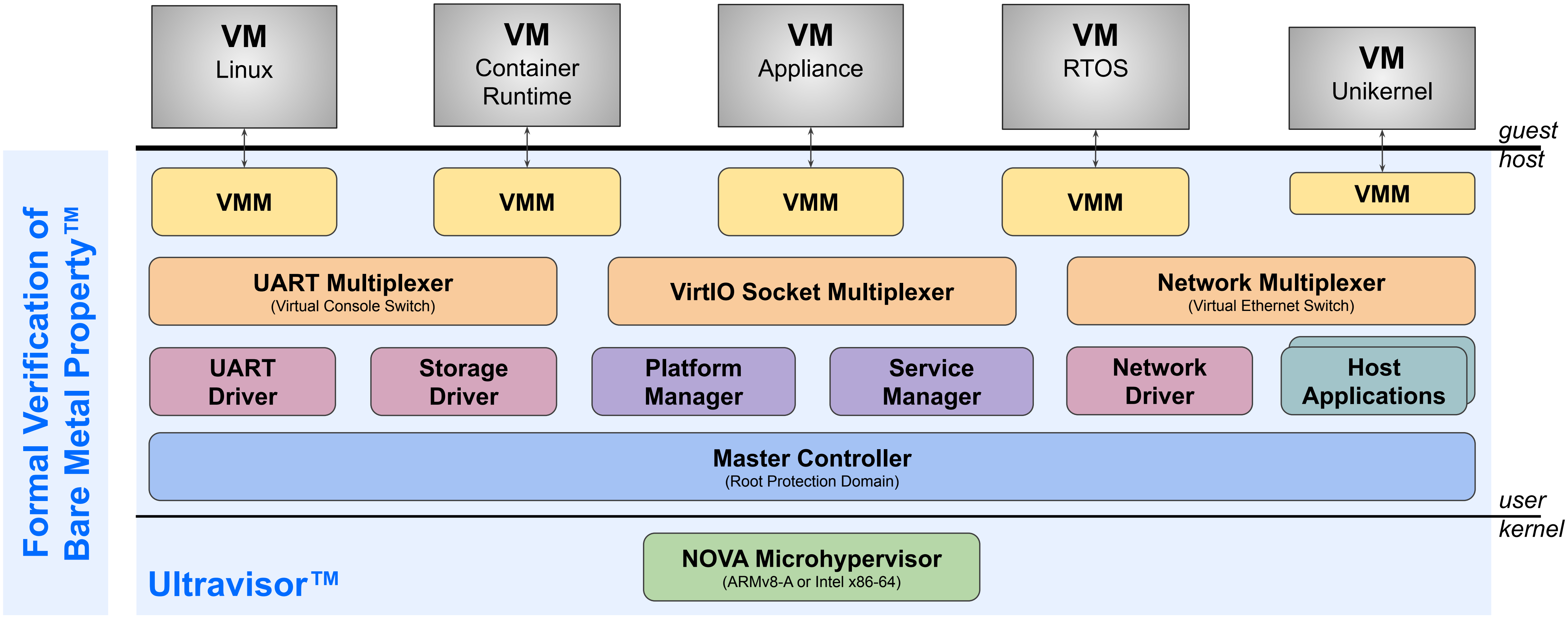
⇒ NOVA implements only mechanisms, but no policies

Capability-Based Access Control



- All syscalls based on capabilities
  - No designation without authority
  - No ambient authority
- Principle of least authority (POLA)
  - Components only possess capabilities for the resources they need
- Capabilities can be delegated
  - Permissions can be further restricted

BedRock Ultravisor Architecture



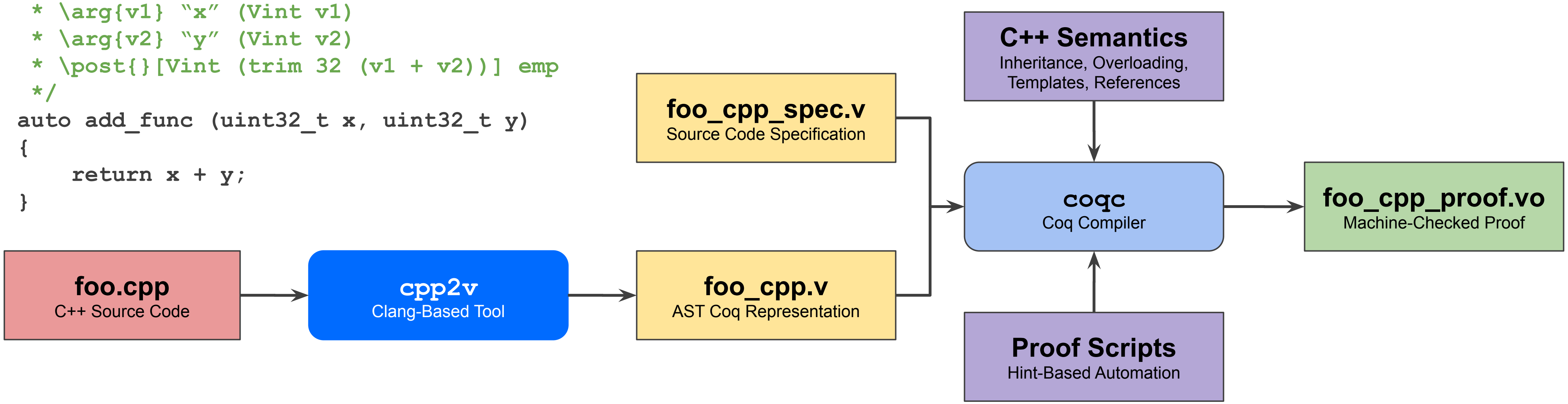
BedRock Systems Inc

Formal Verification: From Source Code to Proof

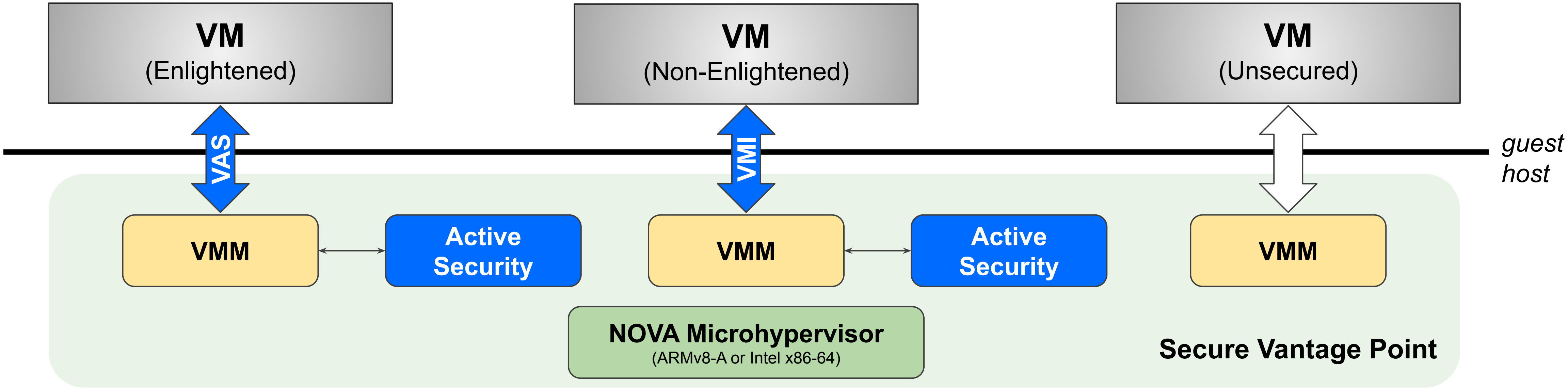
File-Modular Verification of Concurrent C++ Code using Separation Logic

- Specifications can differ for disciplined vs. undisciplined components

```
/*
 * \arg{v1} "x" (Vint v1)
 * \arg{v2} "y" (Vint v2)
 * \post{}[Vint (trim 32 (v1 + v2))] emp
 */
auto add_func (uint32_t x, uint32_t y)
{
    return x + y;
}
```



Active Security: Fortify VMs & Container Runtimes



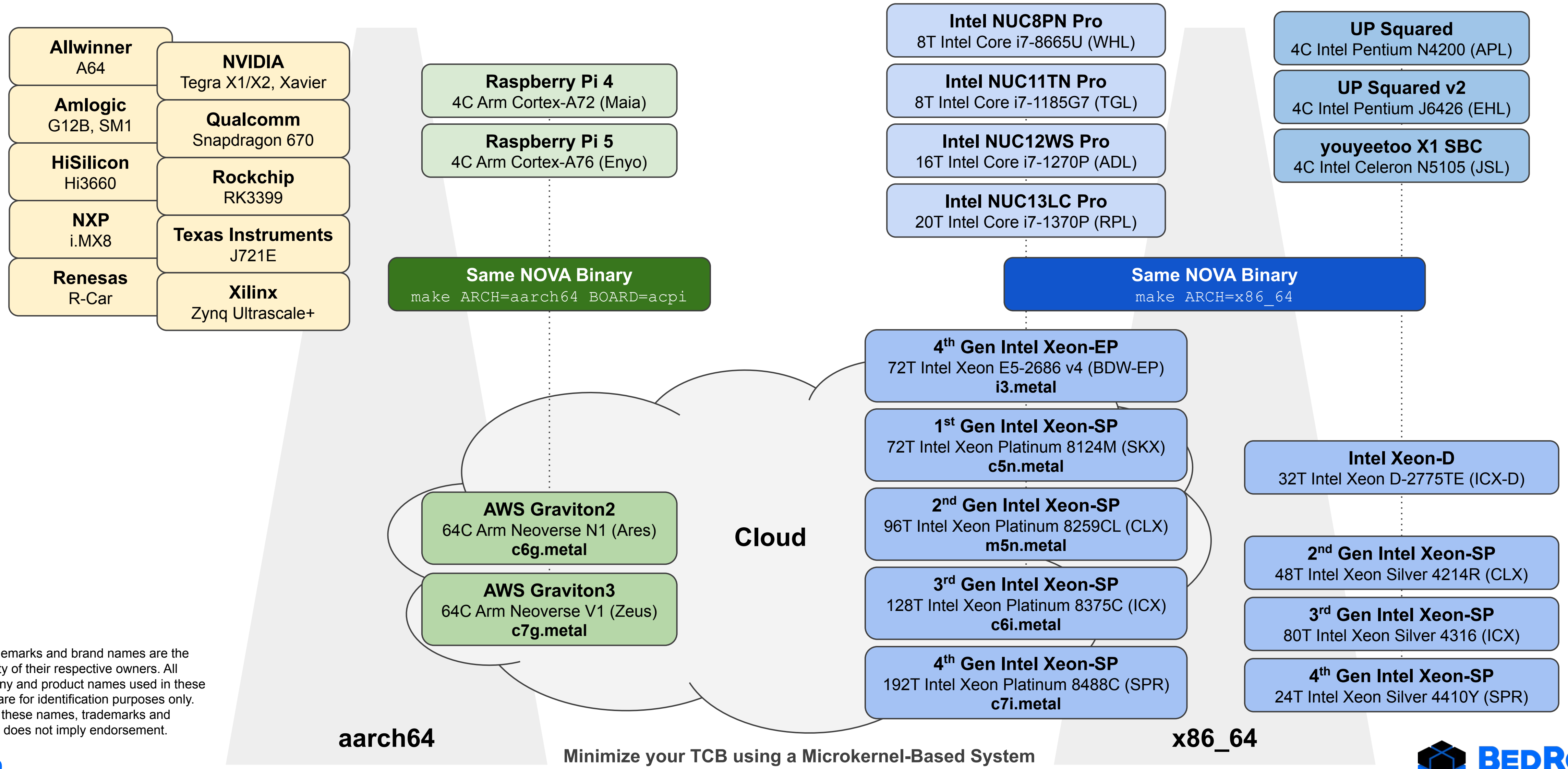
Observe
Non-Bypassable Monitoring

Detect
Invisible Instrumentation

Protect
Software Hardening

Attackers with guest kernel privileges **cannot evade or disarm** the active security mechanisms implemented in the imperceptible Ultravisor layer

Scaling NOVA from Embedded to Cloud Servers

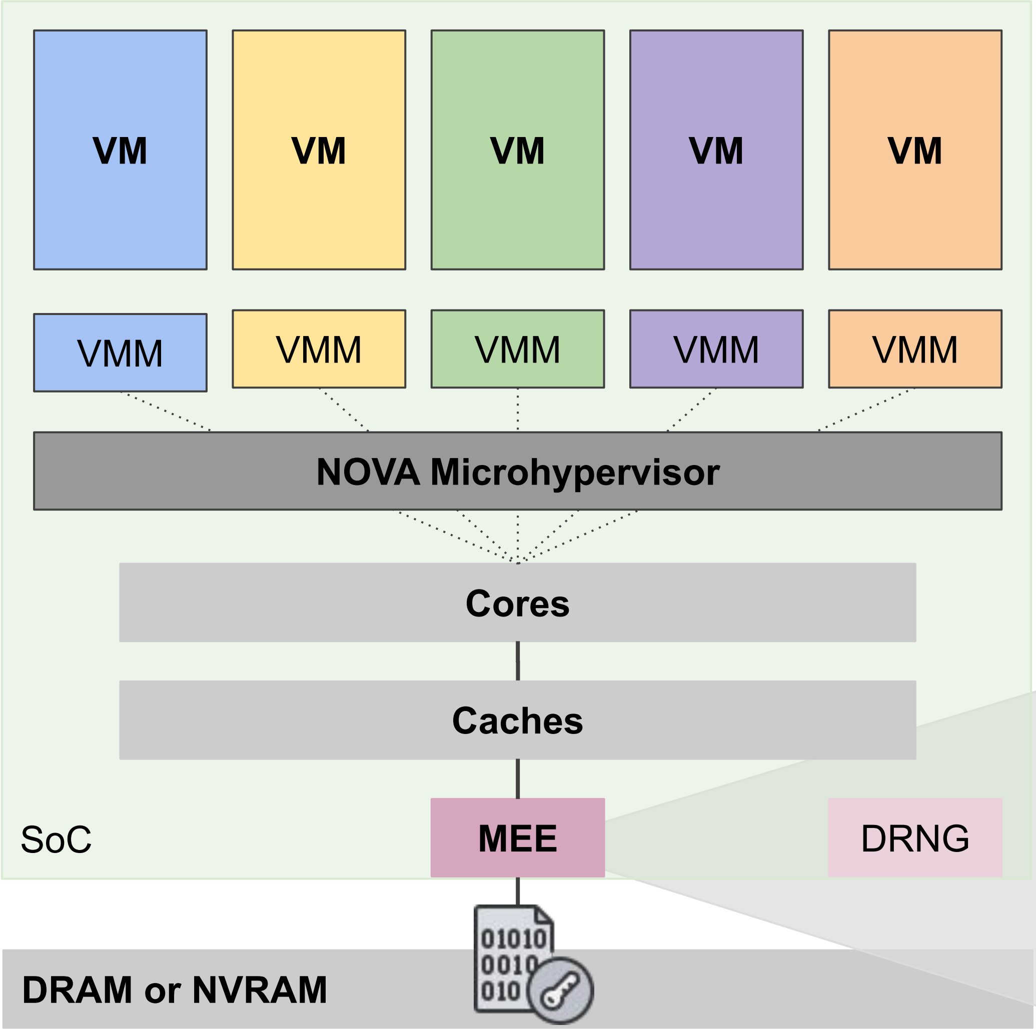


BedRock
Systems Inc

Advanced x86 Security Technologies

Hardening the Platform Further

Multi-Key Total Memory Encryption (TME-MK)



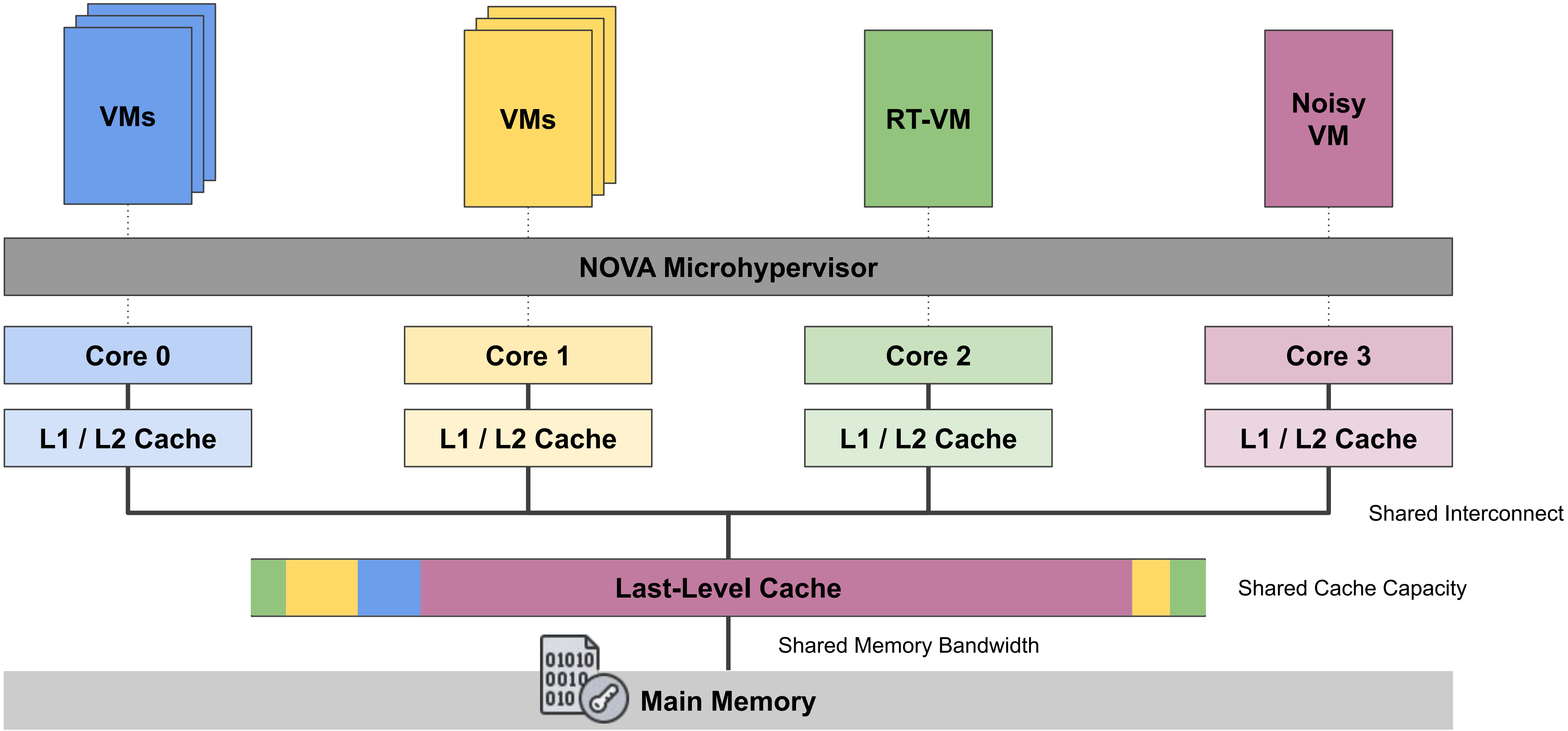
- KeyID per page encoded in PTE
- Stealing upper physical bits

Unused	**KeyID**	**Physical Address**	Attributes
Unused	**KeyID**	**Physical Address**	Attributes
Unused	**KeyID**	**Physical Address**	Attributes
Unused	**KeyID**	**Physical Address**	Attributes

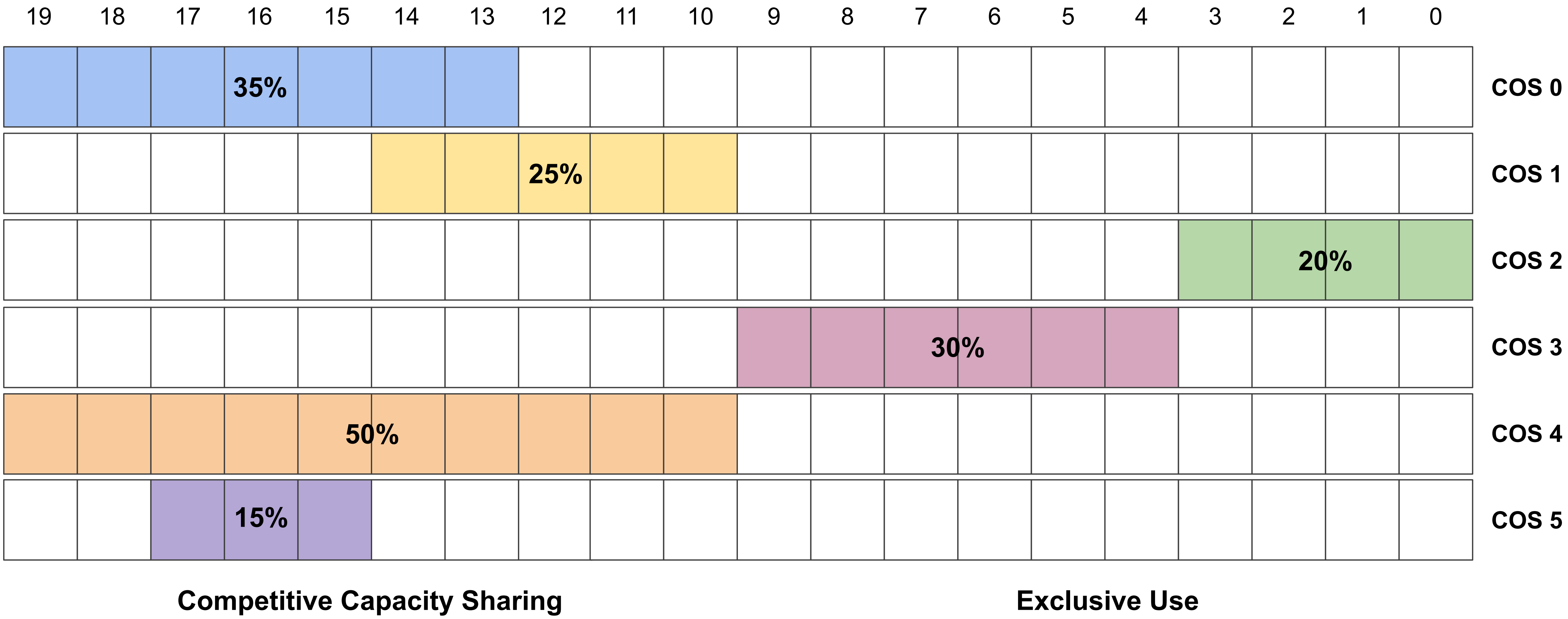
Key0	FW TME Key
Key1	AES-XTS-128
Key2	AES-XTS-256
Key3	AES-XTS-256
Key4	AES-XTS-128
Key5	AES-XTS-128

- Key Programming
  - random/tenant
  - DRNG entropy

Protecting against "Noisy Neighbor" Domains



Cache Allocation Technology (CAT/CDP)



Code Integrity Protection

- Long history of paging features raising the bar for code injection attacks
  - Non-writable code / Non-executable stack (W^X)
  - Supervisor Mode Execution Prevention (SMEP)
  - Supervisor Mode Access Prevention (SMAP)
  - Mode-Based Execution Control (MBEC) for Stage-2 with XU/XS permission bits
- Code snippets (gadgets) in existing code could still be chained together
  - Control-Flow Hijacking: COP / JOP / ROP attacks
  - Instruction length is fixed on ARM but varies on x86

Control-Flow Enforcement Technology (CET)

- Protects integrity of control-flow graph using x86 hardware features
- Indirect Branch Tracking (Forward-Edge) `make ARCH=x86_64 CFP=branch`
  - Used with indirect JMP / CALL instructions
  - Valid branch targets must be marked with ENDBR instruction
  - Requires compiler support (available since gcc-8)
- Shadow Stacks (Backward-Edge) `make ARCH=x86_64 CFP=return`
  - Used with CALL / RET instructions
  - Second stack used exclusively for return addresses
  - Can only be written by control-transfer and shadow-stack-management instructions

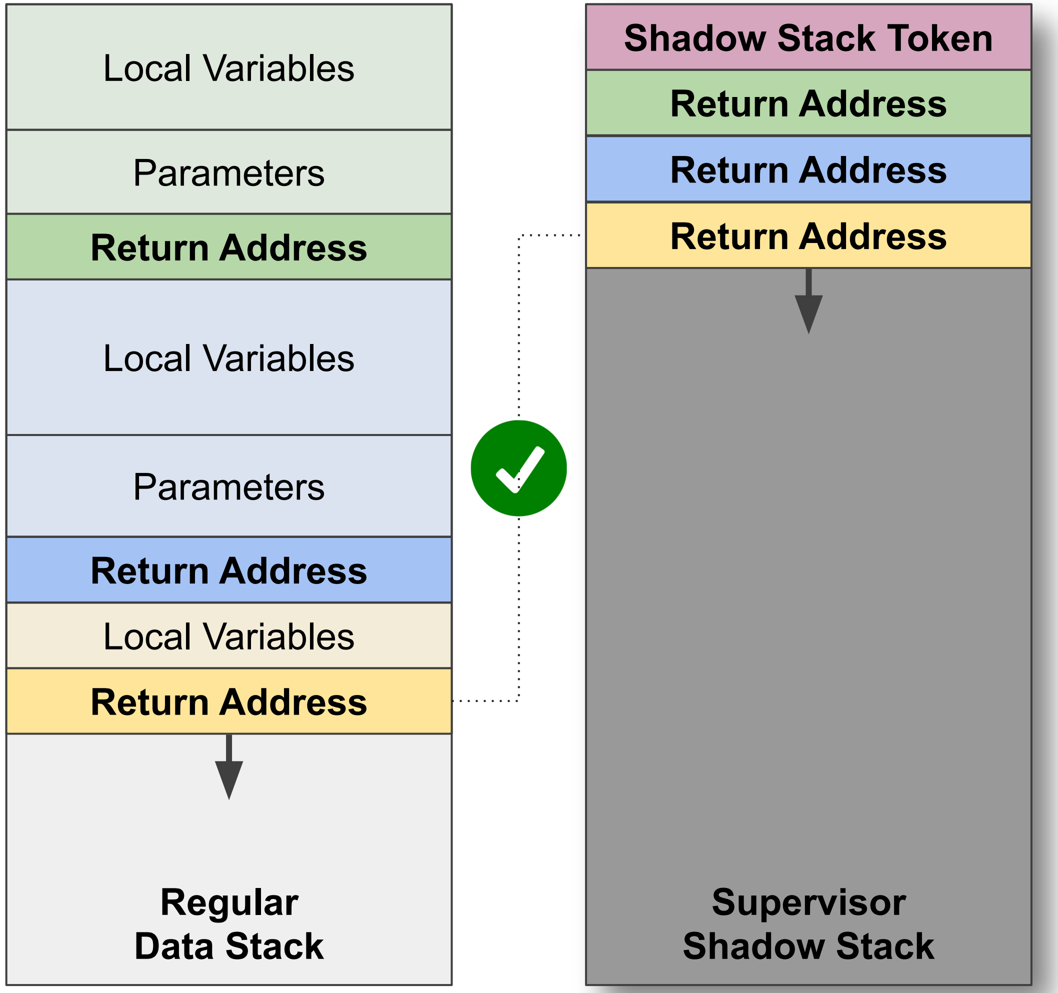
CET Indirect Branch Tracking

```
call *0x30(%rbx)
```

```
ffffffff80003a60 <Buddy::free(void*)>:
ffffffff80003a60:        endbr64
ffffffff80003a64:        test   %rdi,%rdi
ffffffff80003a67:        je     ffffffff80003a84
ffffffff80003a69:        sub    0xf1e8(%rip),%rdi
ffffffff80003a70:        shr    $0xc,%rdi
ffffffff80003a74:        imul   $0x18,%rdi,%rdi
ffffffff80003a78:        add    0xf1d1(%rip),%rdi
ffffffff80003a7f:        jmp    ffffffff80003962
ffffffff80003a84:        ret
```

- CALL / JMP Instruction
  - Next instruction must be ENDBR
  - #CP exception otherwise

CET Supervisor Shadow Stacks



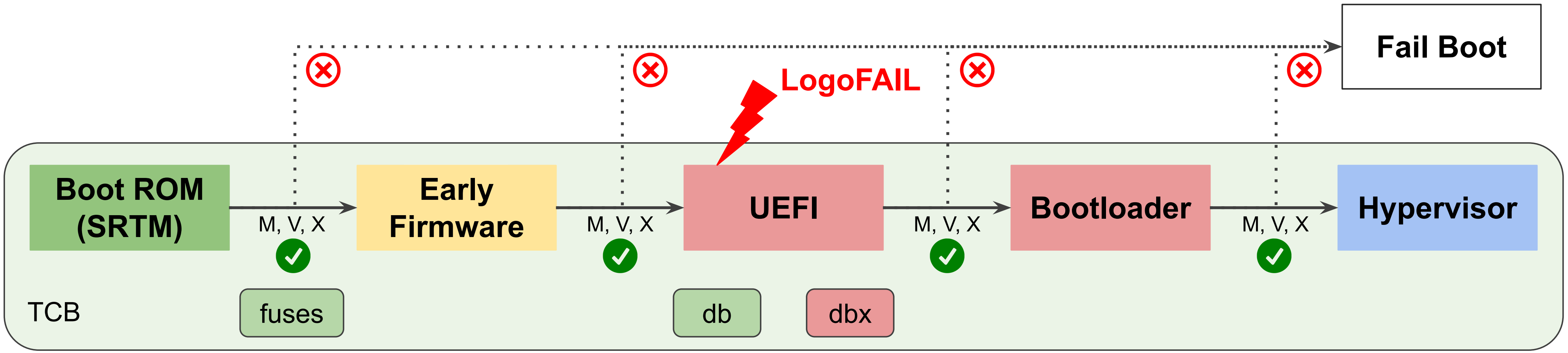
- CALL instruction
  - Pushes return address onto both stacks
- RET instruction
  - Pops return address from both stacks
  - #CP exception if addresses not equal
- Shadow Stack Management
  - Busy bit in token prevents multi-activation
  - NOVA must unwind supervisor shadow stack during context switches

BedRock Systems Inc

Trusted Computing

- Once you have a formally verified software stack
  - and a compiler that produced a qualified set of binaries for the target architecture
- How do you ensure that some computer is running **those** binaries
  - and not some other (malicious) software instead
  - before you entrust that computer with your data or secrets
- In other words, how can you
  - either restrict the software that a computer will launch
  - or determine what software has been launched on a computer

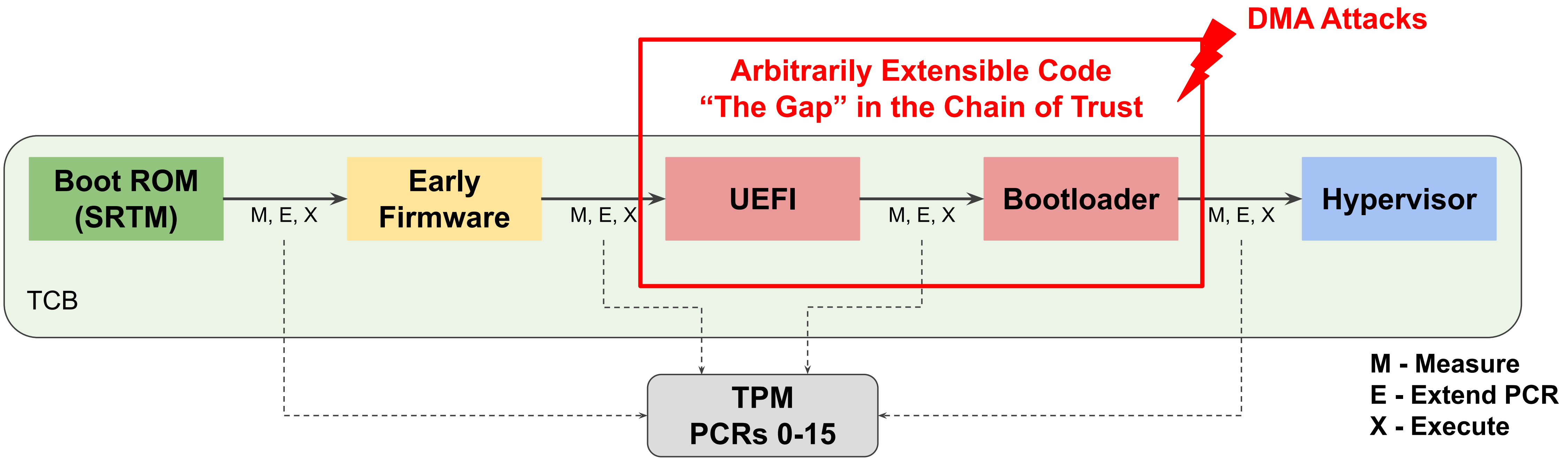
Verified Boot: Static Root of Trust



M - Measure
V - Verify
X - Execute

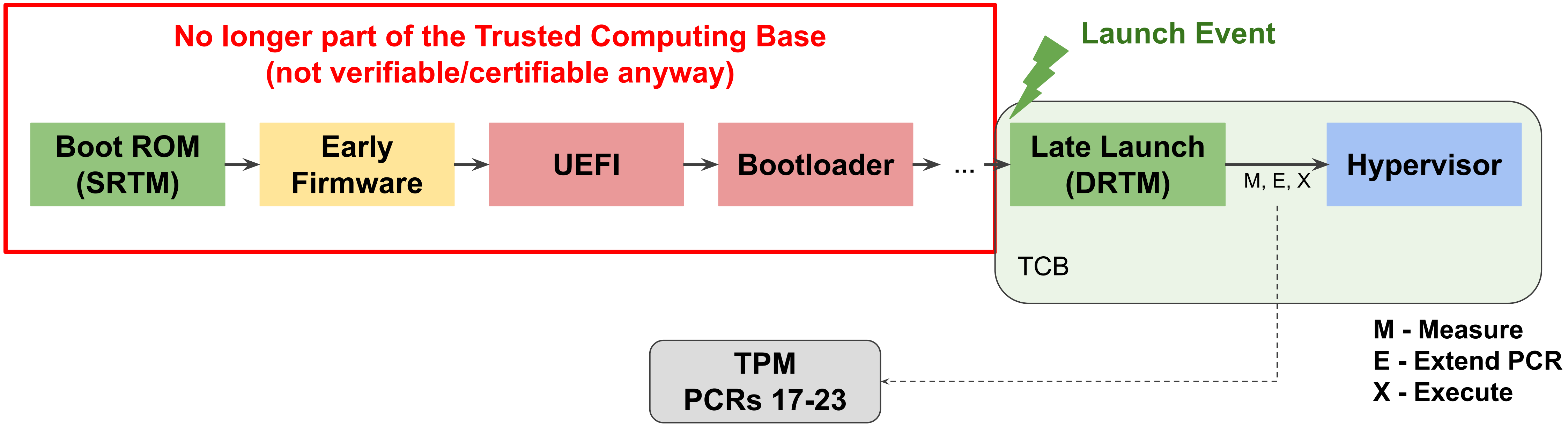
- Boot policies are enforced during the boot process
- Starting with the Core Root of Trust for Verification, the currently executing module verifies the integrity of the next module against a boot policy (e.g. UEFI db/dbx) ⇒ Chain of Trust
- Integrity measurement is a cryptographic hash ⇒ unique + indicative to changes in the module

Measured Boot: Static Root of Trust



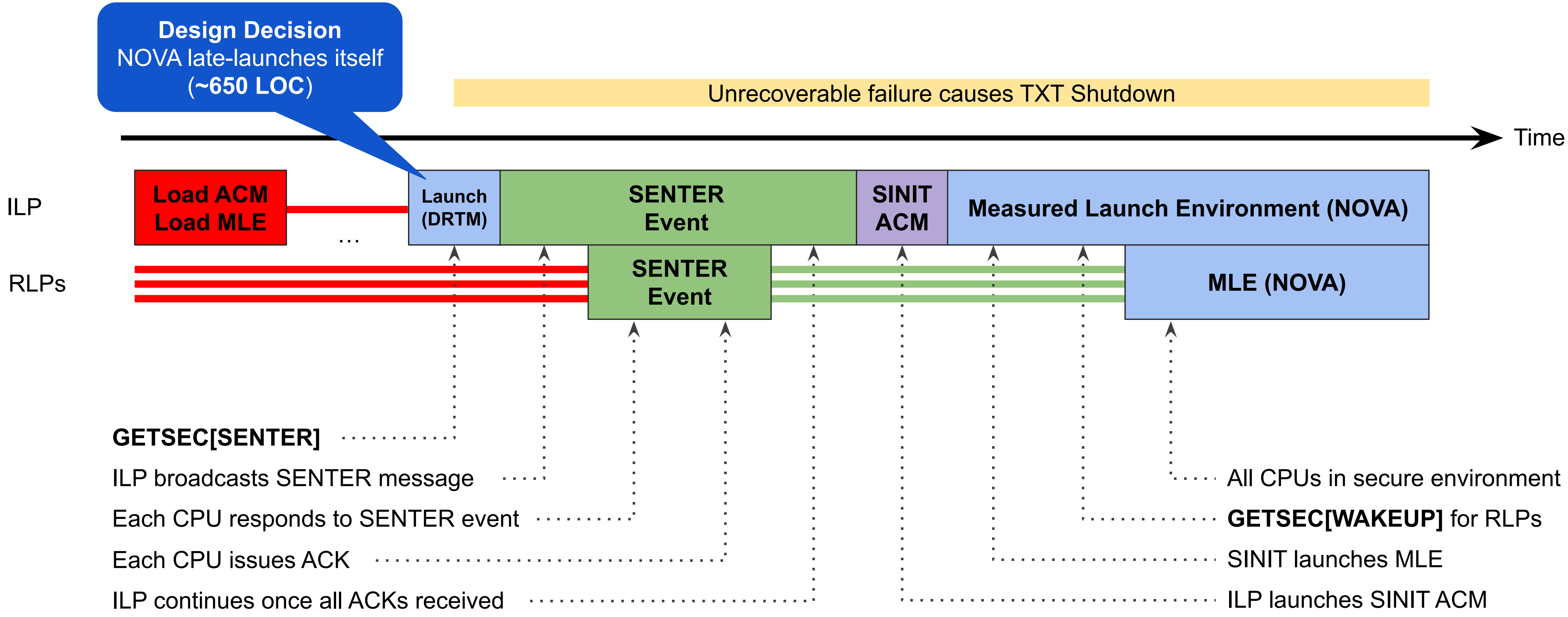
- Integrity measurements are extended into TPM PCRs during the boot process
- Starting with the Core Root of Trust for Measurement, the currently executing module extends the launch integrity measurement for the next module into the TPM

Measured Boot: Dynamic Root of Trust



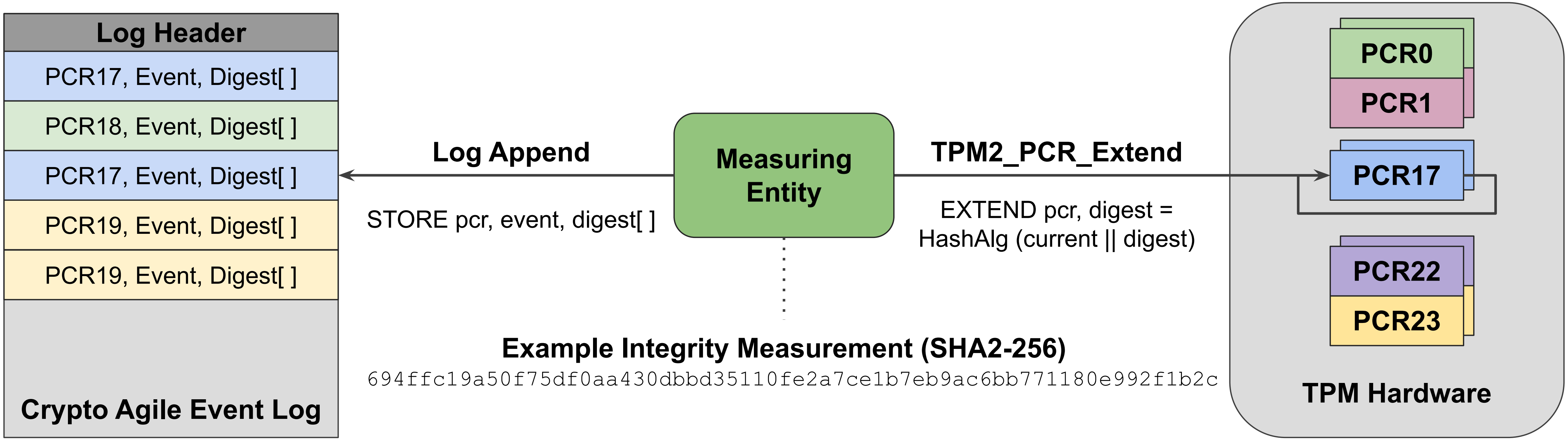
- DRTM Flow lets system boot into an untrustworthy state (initially)
  - Measured Launch later “resets” system into a trustworthy safe state
  - Takes control of all CPUs and forces them down a <u>protected</u> and <u>measured</u> code path

Trusted Execution Technology: Measured Launch



Trusted Platform Module (TPM)

Image Source:
Infineon Technologies



A verifier can use the crypto agile event log to recompute/validate the composite value in each PCR

Confidential & Trusted Computing Building Blocks

- **Availability**
  - Cache & Memory Bandwidth Allocation Technology (CAT/CDP/MBA)
- **Integrity**
  - Control-Flow Enforcement Technology (CET IBT+SSS)
- **Confidentiality**
  - Total Memory Encryption with Multiple Keys (TME-MK)
- **Measured Launch & Attestation**
  - Trusted Execution Technology (TXT/CBnT)

Questions and Discussion

The NOVA microhypervisor is licensed under GPLv2

Releases: https://github.com/udosteinberg/NOVA/tags

More Information: bedrocksystems.com and hypervisor.org