School of Computer Science & Engineering

Trustworthy Systems Group

The seL4® Microkernel

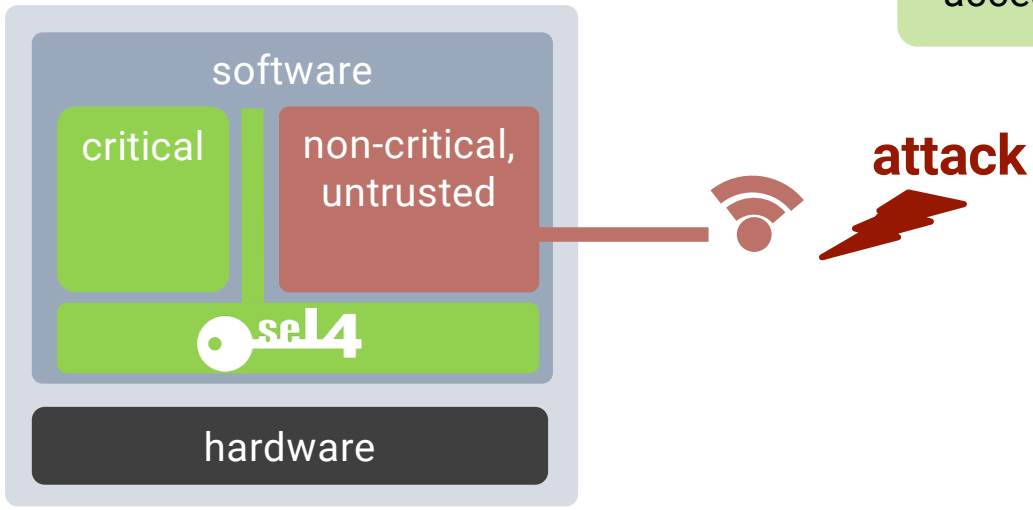
Mathematical Proof of Security

Gernot Heiser, UNSW & seL4 Foundation
@GernotHeiser

Background: What is seL4?

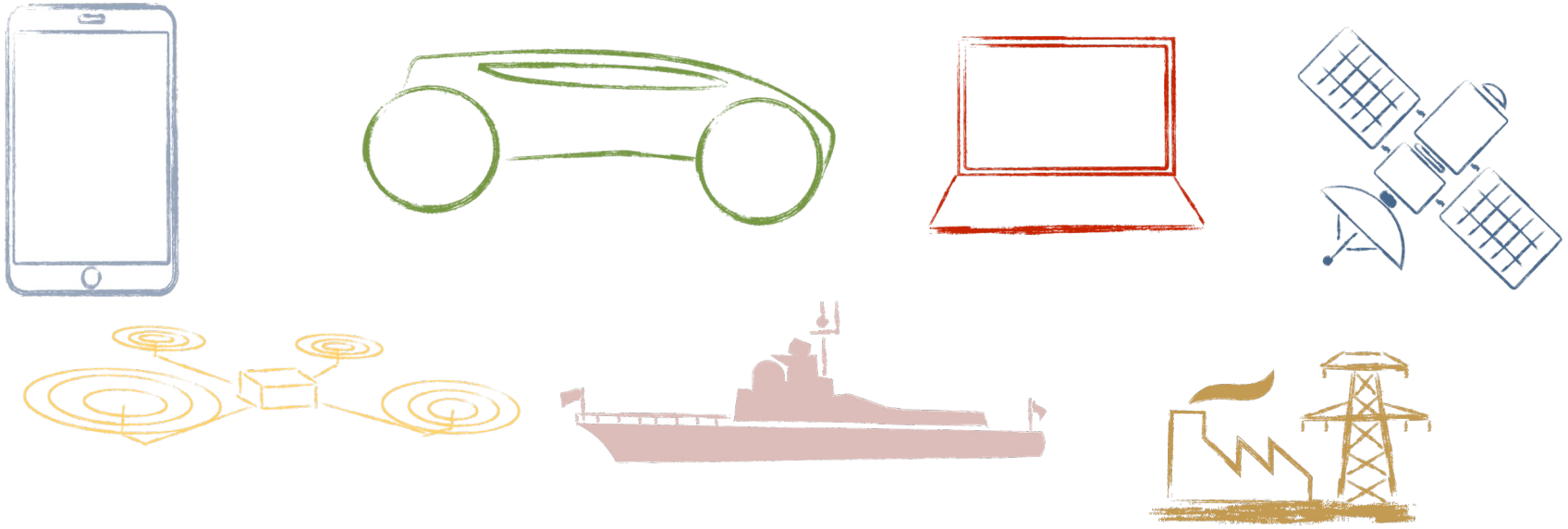
seL4 is an open source, high-assurance, high-performance operating system microkernel

- Available on GitHub under GPLv2 license (code and proofs!)
- World's most comprehensive mathematical proofs of correctness and security
- World's fastest microkernel
- Piece of software that runs at the heart of any system and controls all accesses to resources



What is seL4?

seL4 is the most trustworthy foundation for safety- and security-critical systems



Deployed / in designs across many domains:
automotive, avionics, space, defence, IoT, industry 4.0

The Benchmark for Performance

Latency (in cycles) of a round-trip cross-address-space IPC on x64

Source	seL4	Fisco.OC	Zircon
Mi et al, 2019	986	2717	8157
Gu et al, 2020	1450	3057	8151
seL4.systems, Feb'21	814	N/A	N/A

World's fastest microkernel!

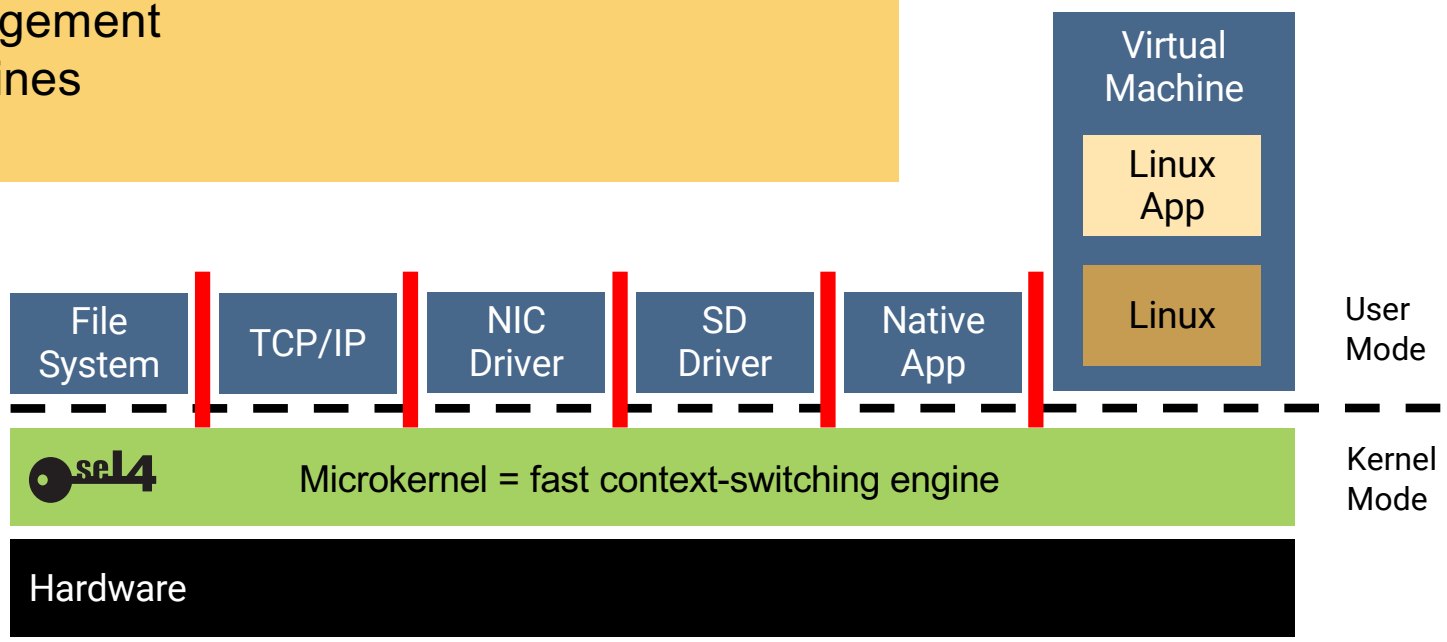
Sources:

- Zeyu Mi, Dingji Li, Zihan Yang, Xinran Wang, Haibo Chen: "SkyBridge: Fast and Secure Inter-Process Communication for Microkernels", EuroSys, April 2020
- Jinyu Gu, Xinyue Wu, Wentai Li, Nian Liu, Zeyu Mi, Yubin Xia, Haibo Chen: "Harmonizing Performance and Isolation in Microkernels with Efficient Intra-kernel Isolation and Communication", Usenix ATC, June 2020
- seL4 Performance, https://sel4.systems/About/Performance/, accessed 2020-11-08

A Microkernel is not an OS

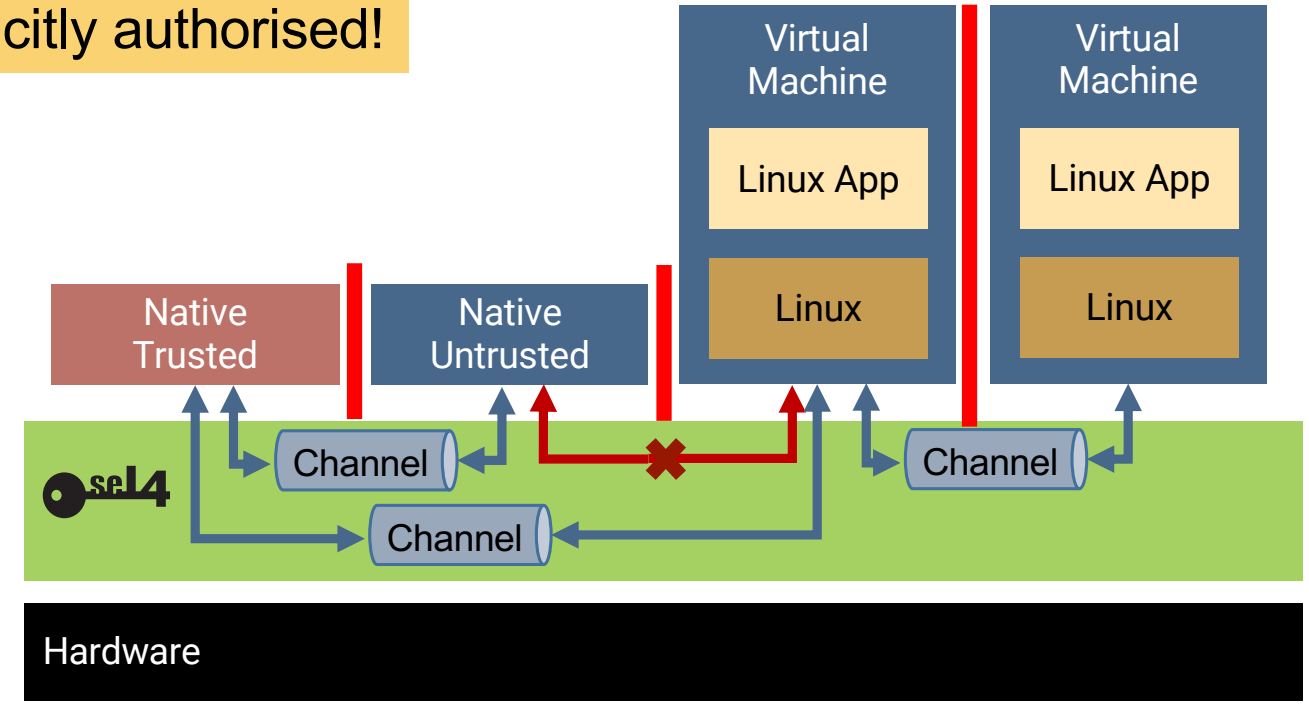
All operating-system services are user-level processes:

- file systems
- device drivers
- power management
- virtual machines
- …

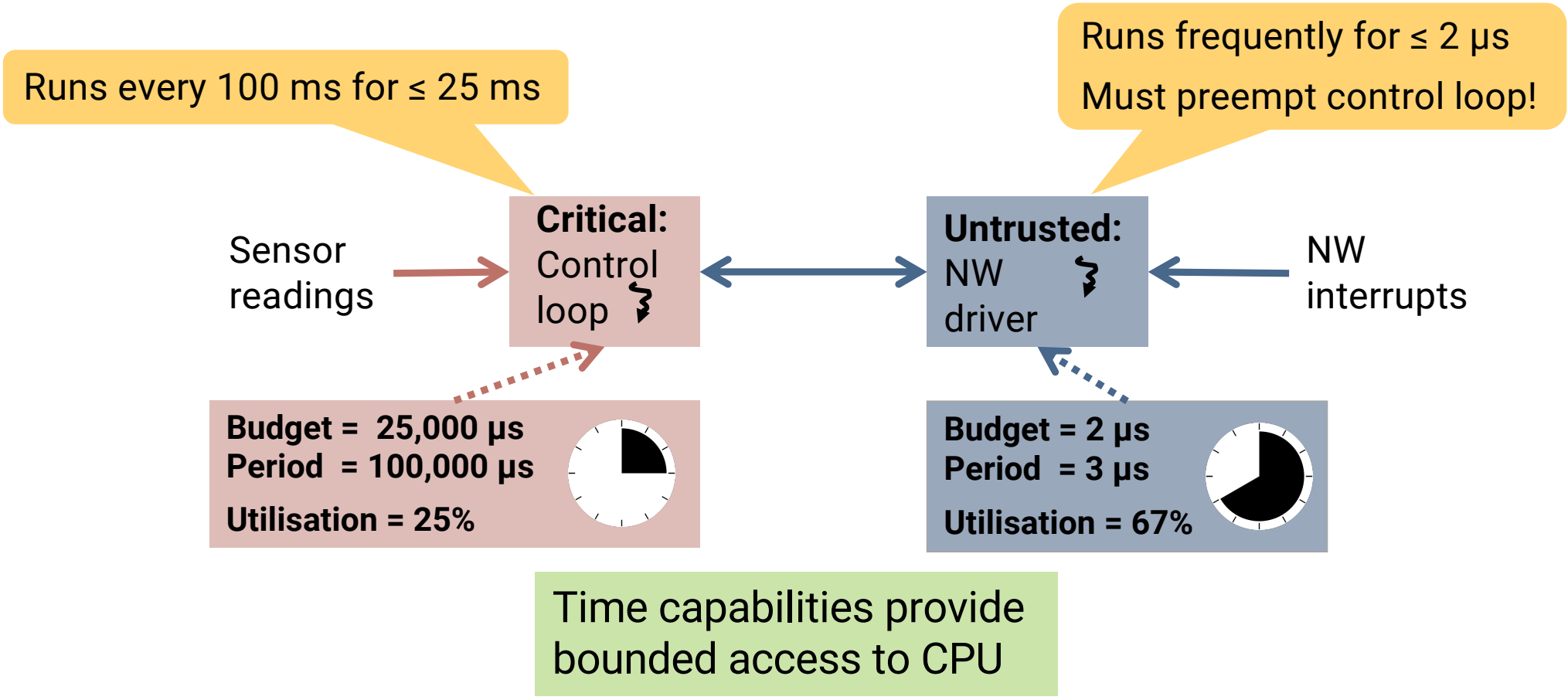


Capabilities Control Communication

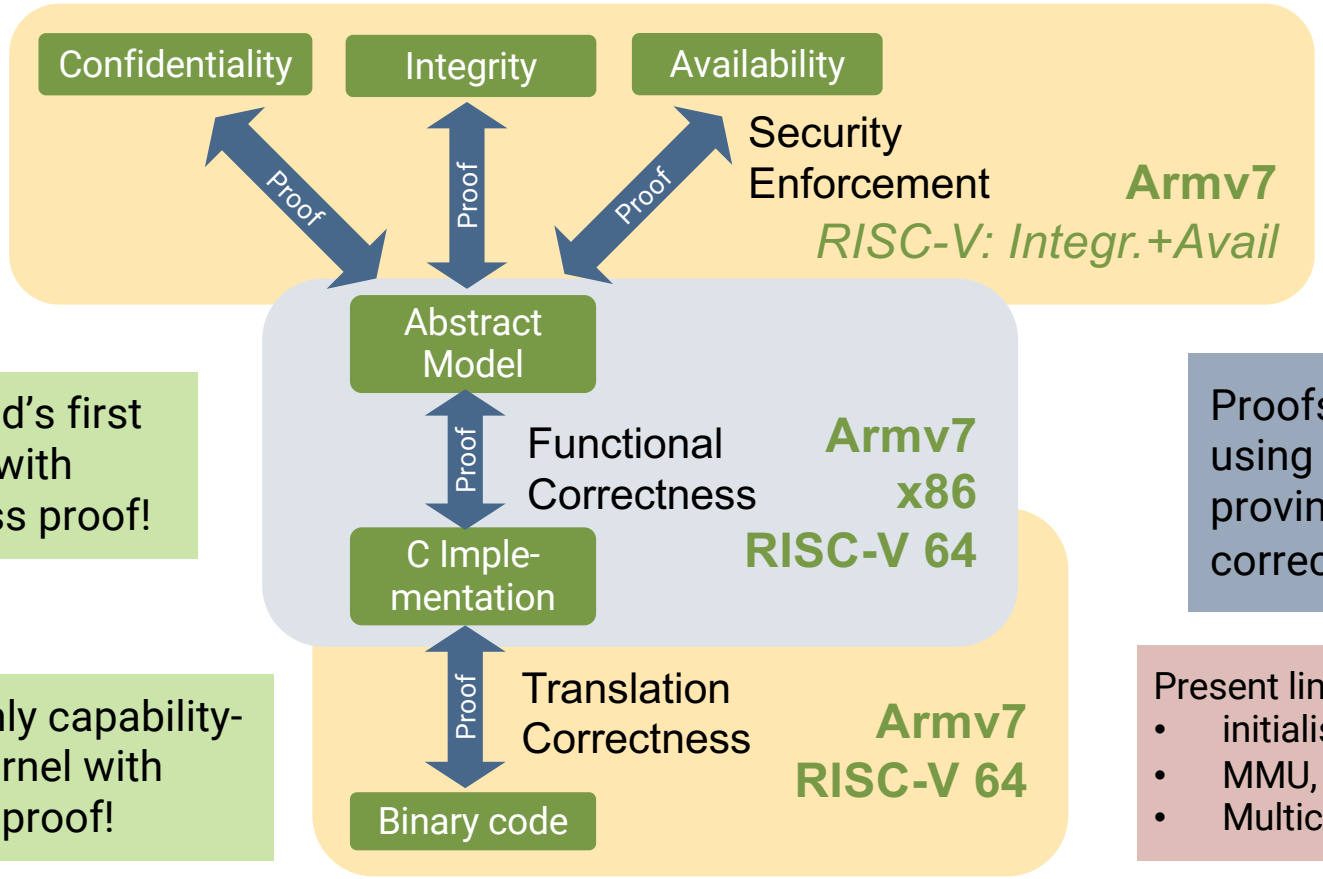
No communication
unless explicitly authorised!



New MCS Kernel: Capabilities for Time



Trustworthiness By Mathematical Proof



seL4: World's first OS kernel with correctness proof!

seL4: Still only capability-based OS kernel with correctness proof!

Proofs are machine-checked, using interactive theorem proving (translation correctness fully automated)

Present limitations
- initialisation code not verified
- MMU, caches modelled abstractly
- Multicore not yet verified

What Does This Mean?

Kinds of properties proved for functional correctness

- Behaviour is fully captured by abstract model
- Kernel never fails, behaviour is always well-defined
  - ✓ assertions never fail
  - ✓ will never de-reference null pointer
  - ✓ will never access array out of bounds
  - ✓ cannot be subverted by mis-formed input
  - ✓ …

Can prove further properties on abstract level!

Military-Grade Security

Cross-Domain Desktop Compositor

Multi-level secure terminal
Commercialisation in progress

Laot: Critical infrastructure protection

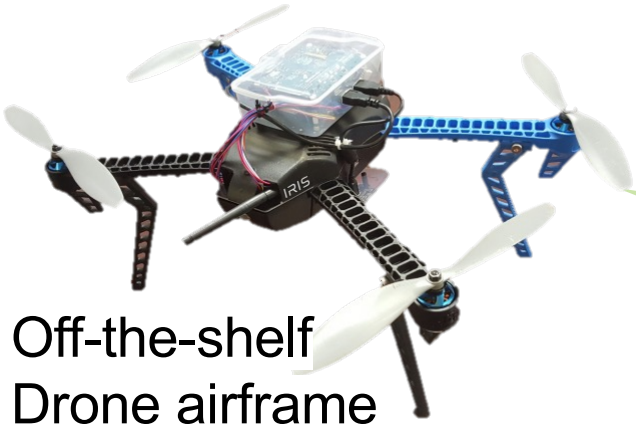
Secure communication device
In use in AU, UK defence forces

Securing Systems: DARPA HACMS

Unmanned Little Bird (ULB)

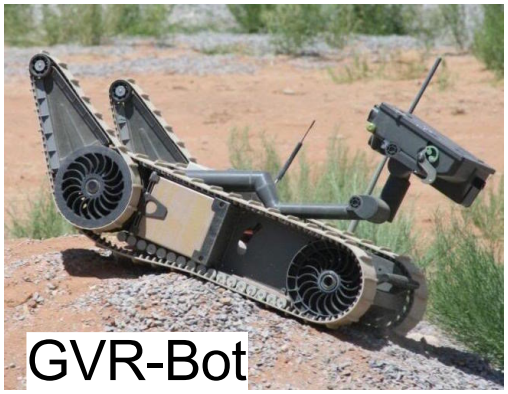
Retrofit existing system!

Autonomous trucks



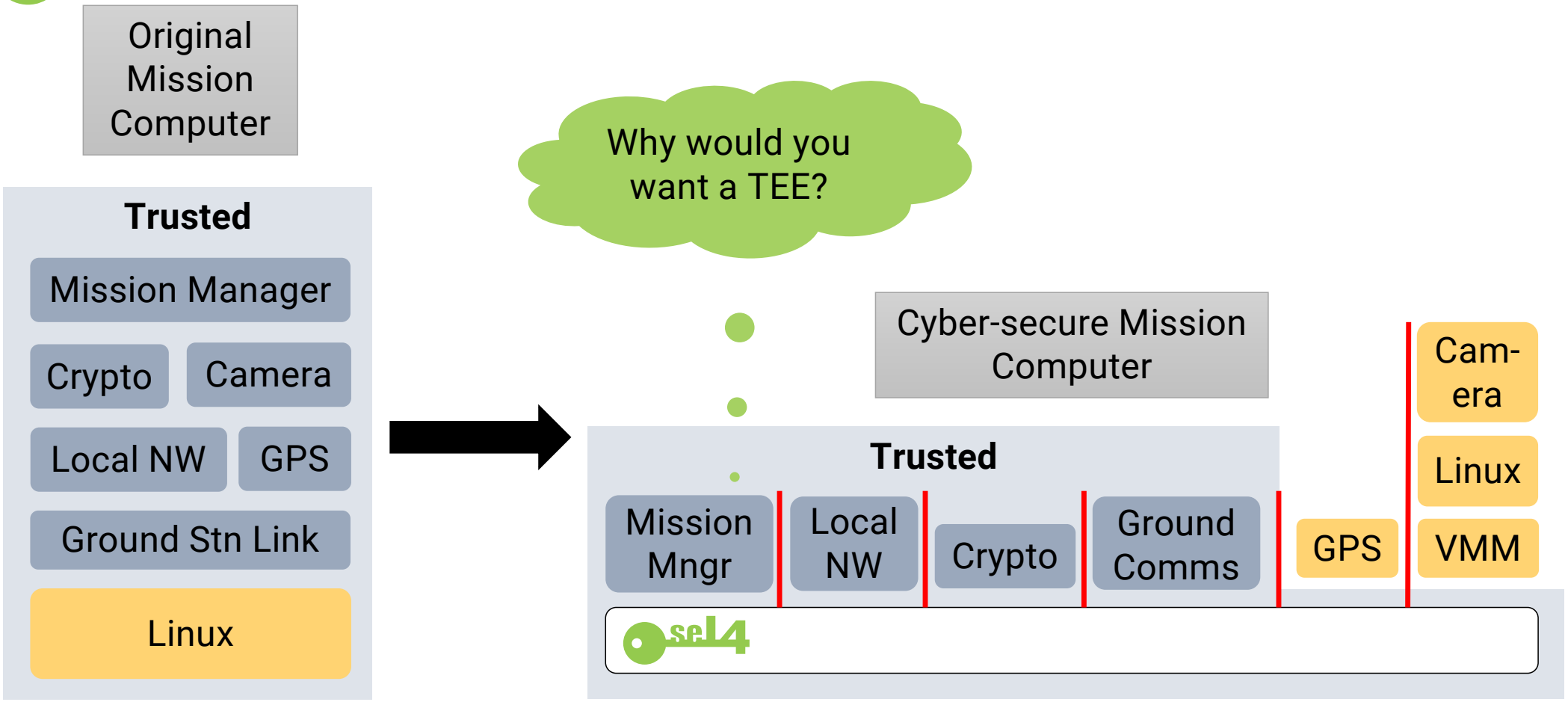
Off-the-shelf Drone airframe

Develop technology



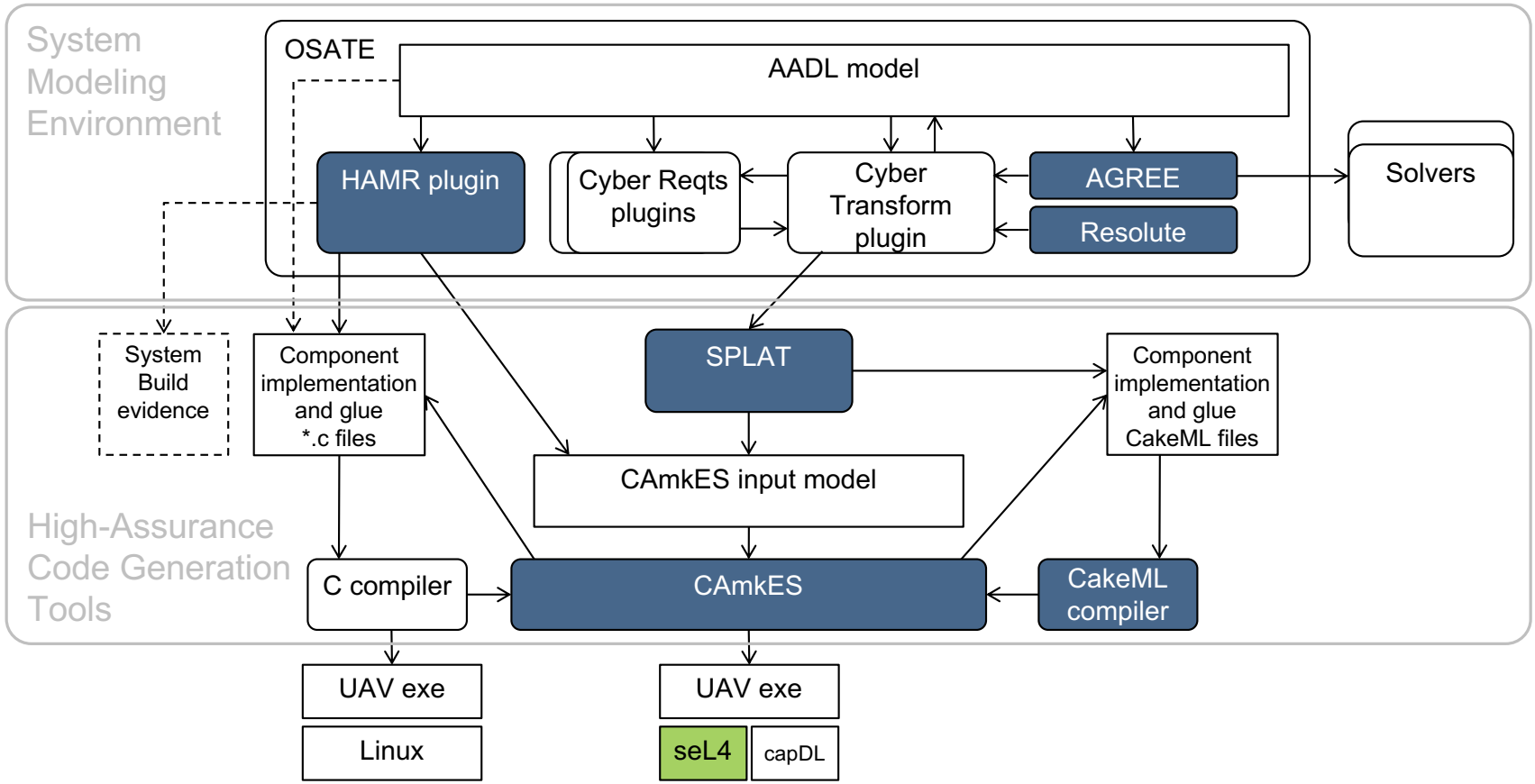
GVR-Bot

Incremental Cyber Retrofit

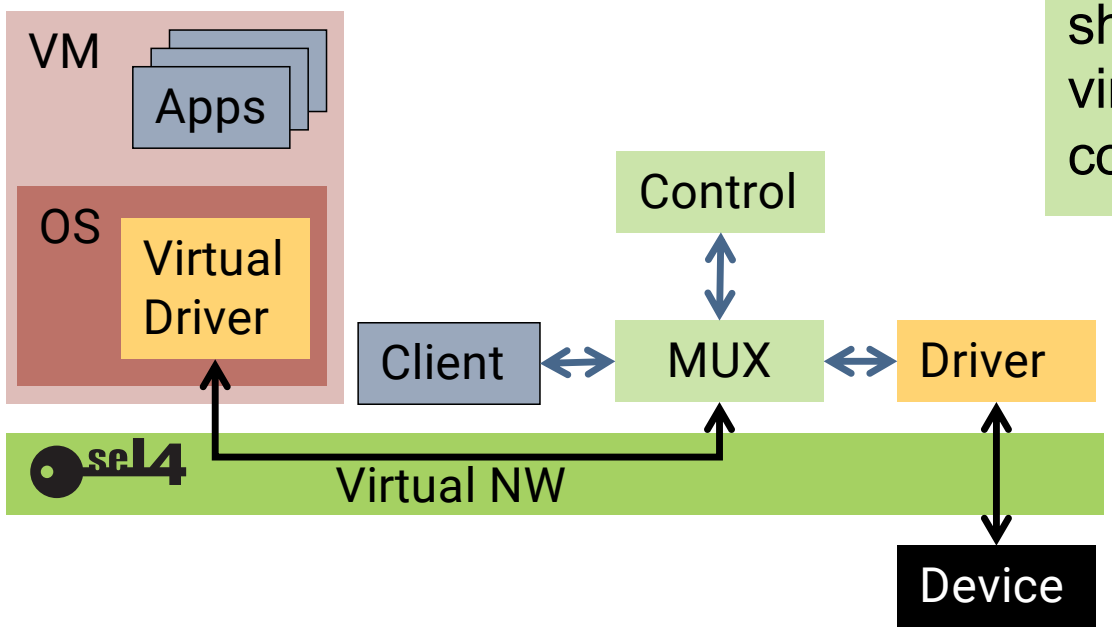


Present Projects

DARPA CASE: Repeatable Engineering



Secure Device Virtualisation

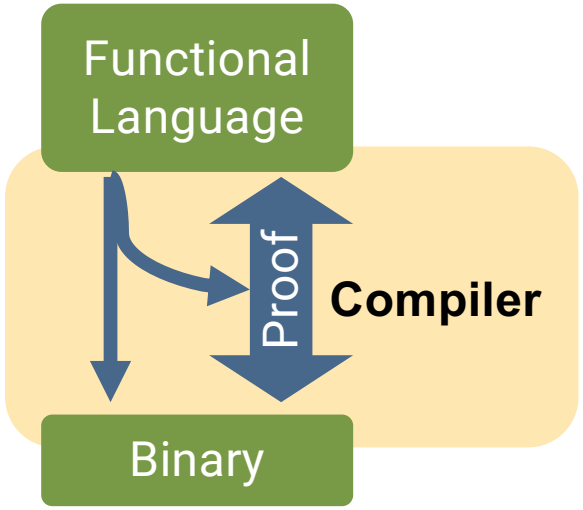


Aim: Secure, low-overhead sharing of devices between virtual machines and native components

Explore verification of critical parts (MUX, Control, Drivers?)

Reducing Cost of Verified Systems SW

Aim: Reduce cost of verified systems code

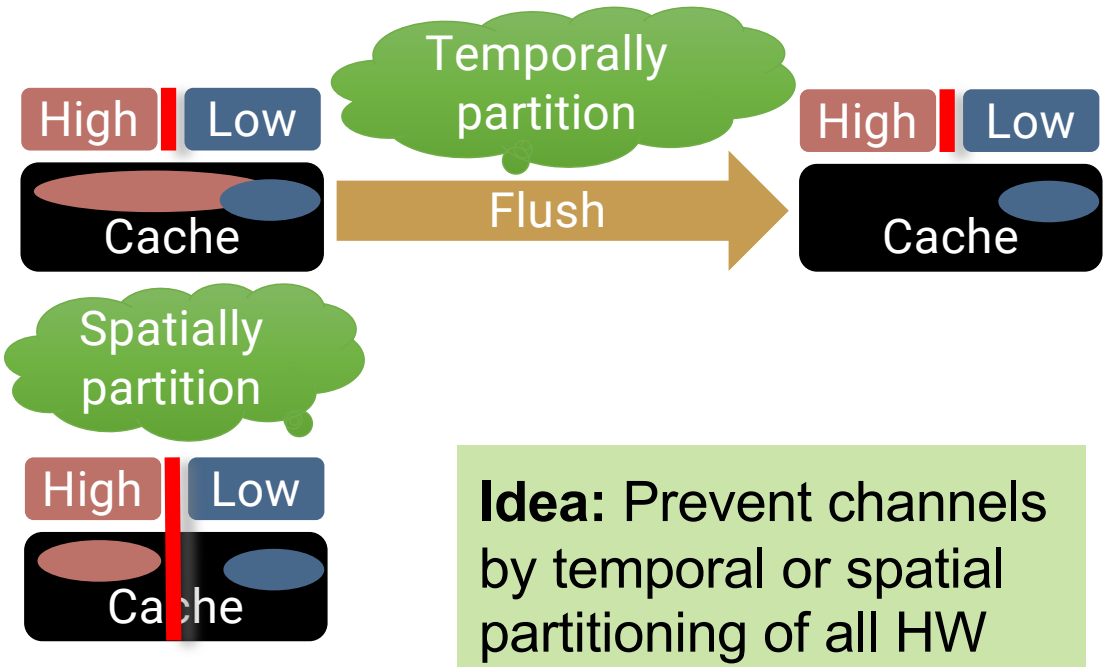


Idea: Use high-level ***systems*** language with certifying compiler

Systems language:

- functional, type & memory safe
- not managed
- low-level (obvious translation)
- interfacing to hardware
- minimal run time

Time Protection: Timing-Channel Prevention

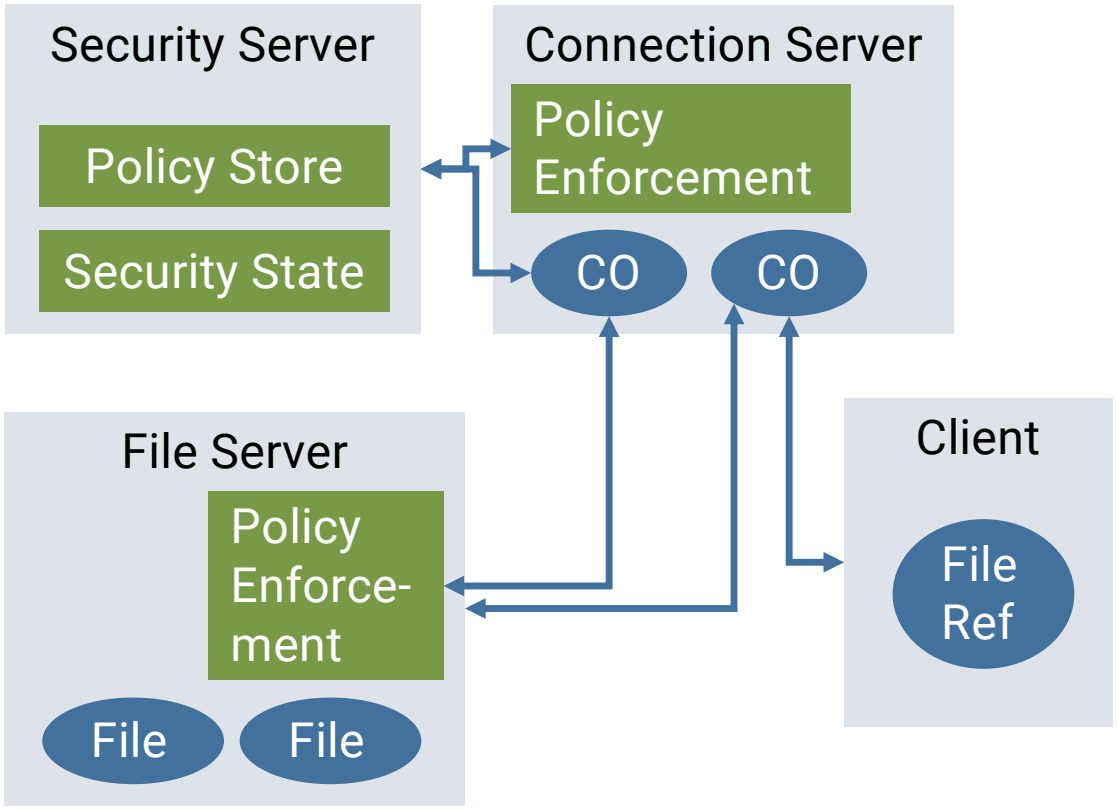


Aim: *Provably* prevent information flow through micro-architectural timing channels

Idea: Prevent channels by temporal or spatial partitioning of all HW

Observation: Timing channels result from competition for limited hardware resources

Secure, General-Purpose OS



Aim: General-purpose OS that *provably* enforces a security policy

Requires:

- mandatory policy enforcement
- policy diversity
- minimal TCB
- low-overhead enforcement

seL4 Foundation

Premium Members

General Members

Associate Members

R&D Funding Strategy

- Close verification gaps:
  - complete RISC-V verification stack (hypervisor, CapDL, …)
  - complete MCS kernel verification (advanced real-time features)
  - verify AArch64 kernel
  - verify multicore kernel
- Develop complete automotive/autonomy/IoT OS
  - open-source
  - based on seL4 Core Platform
  - supported by industry-quality tooling
  - verify critical components

Summary

- Mathematical proof techniques can be applied to real-world software
- Provable security is possible – for a well-designed system
- seL4 is a rock-solid basis for security/safety-critical systems

Defining the state of the art in trustworthy operating systems for over 10 years

Further Reading:

- About seL4: https://sel4.systems/
- seL4 whitepaper: https://sel4.systems/About/seL4-whitepaper.pdf
- seL4 Foundation: https://sel4.systems/Foundation