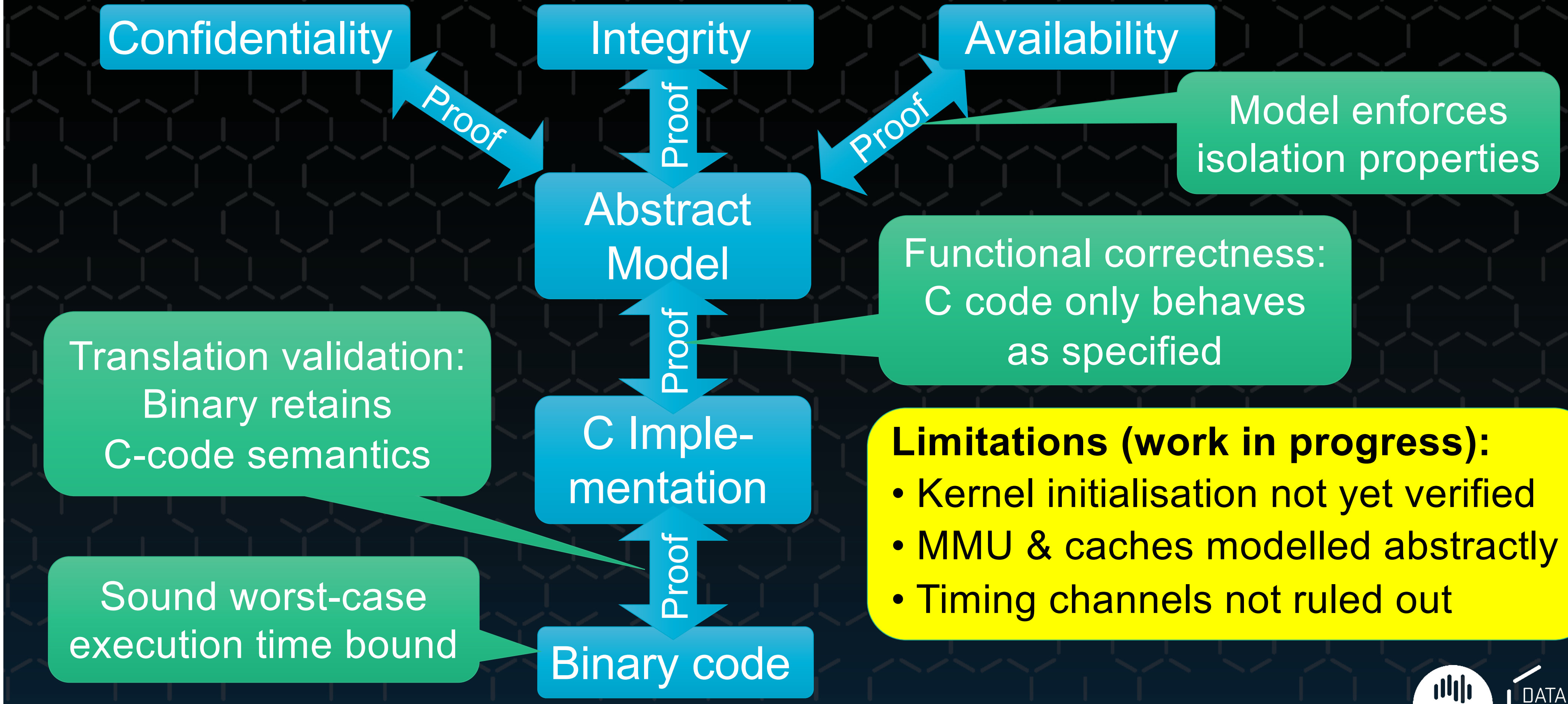
Making the (Software) TCB Trustworthy

Gernot Heiser | gernot@unsw.edu.au | @GernotHeiser

- CPS-VO FMaS, Menlo Park, 9 October 2019

https://trustworthy.systems

seL4: Base for Trustworthy Systems



Model enforces isolation properties

Functional correctness: C code only behaves as specified

Translation validation: Binary retains C-code semantics

Sound worst-case execution time bound

Limitations (work in progress):

- Kernel initialisation not yet verified
- MMU & caches modelled abstractly
- Timing channels not ruled out

Real-World Use: Incremental Cyber Retrofit

DARPA HACMS

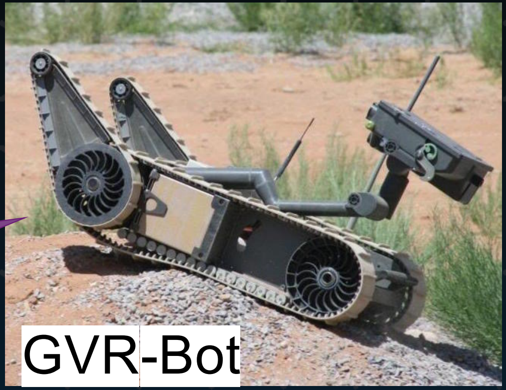
Unmanned Little Bird (ULB)

Retrofit existing system!

Autonomous trucks

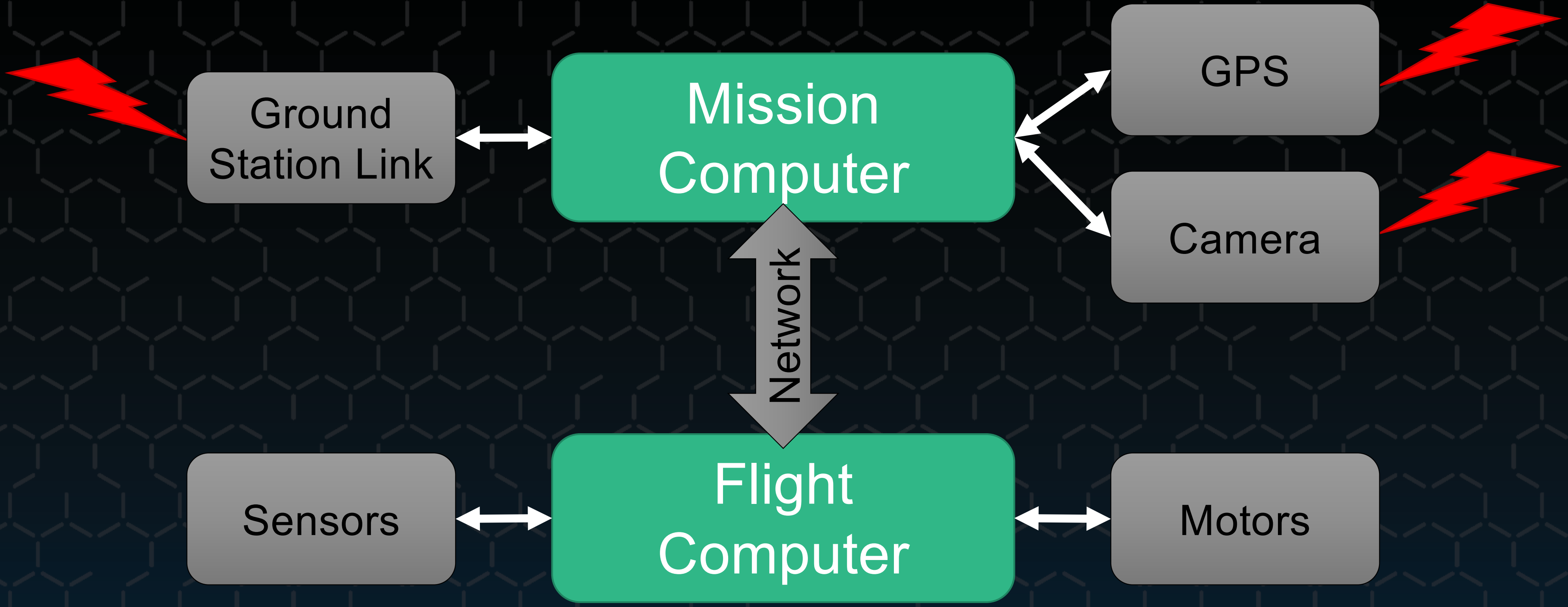
Off-the-shelf Drone airframe

Develop technology

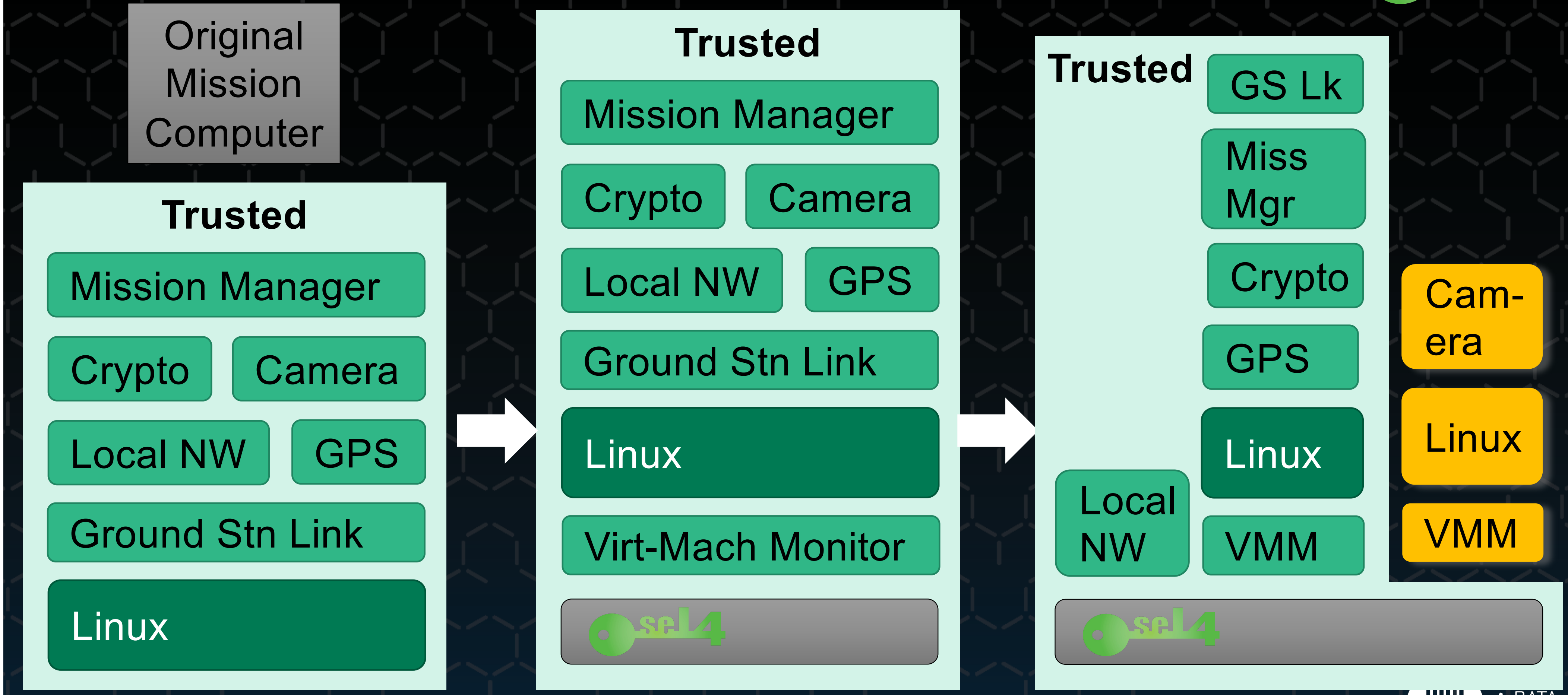


GVR-Bot

ULB Architecture



Incremental Cyber Retrofit



Original Mission Computer

Trusted

Mission Manager

Crypto

Camera

Local NW

GPS

Ground Stn Link

Linux

Trusted

Mission Manager

Crypto

Camera

Local NW

GPS

Ground Stn Link

Linux

Virt-Mach Monitor

seL4

Trusted

GS Lk

Miss Mgr

Crypto

GPS

Linux

Local NW

VMM

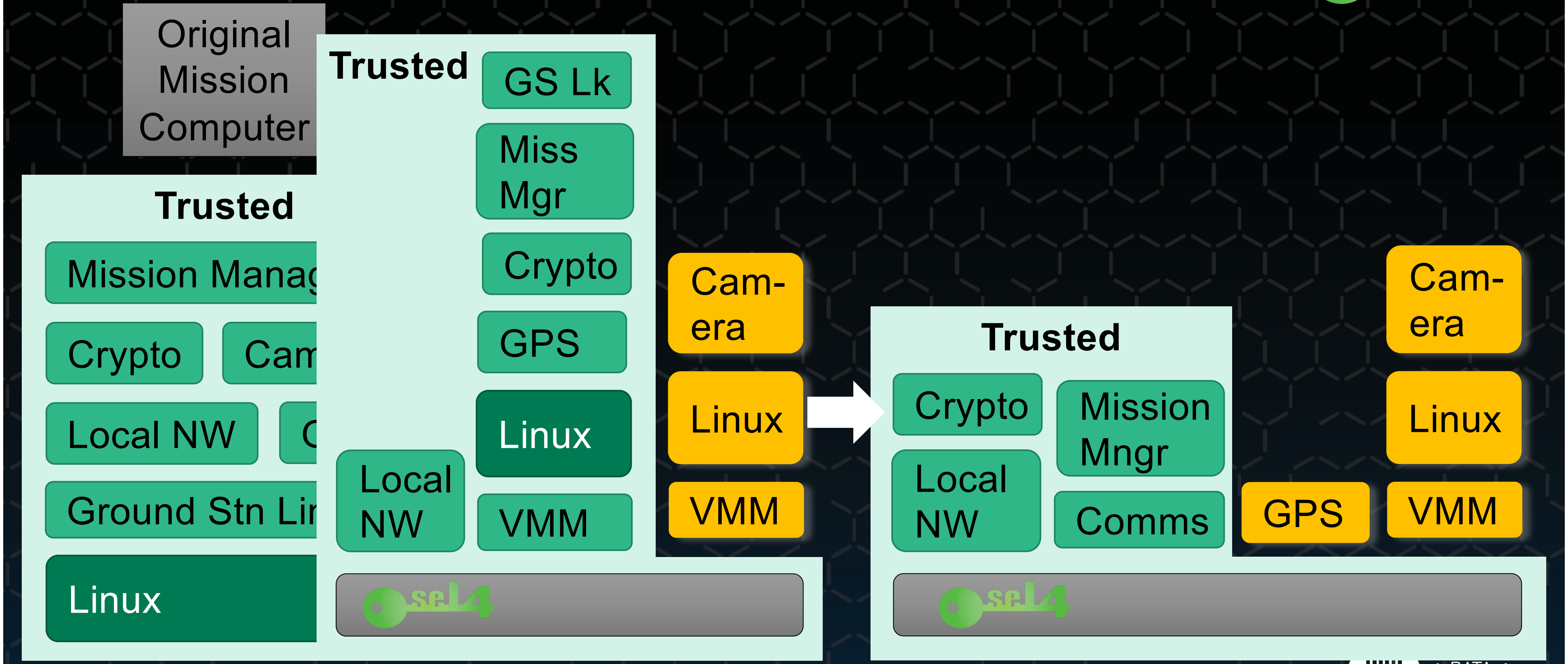
Cam-era

Linux

VMM

seL4

ULB Incremental Cyber Retrofit



Original Mission Computer

Trusted

Mission Manag

Crypto

Cam

Local NW

C

Ground Stn Lir

Linux

Trusted

GS Lk

Miss Mgr

Crypto

GPS

Linux

Local NW

VMM

seL4

Cam-era

Linux

VMM

Trusted

Crypto

Mission Mngr

Local NW

Comms

GPS

Cam-era

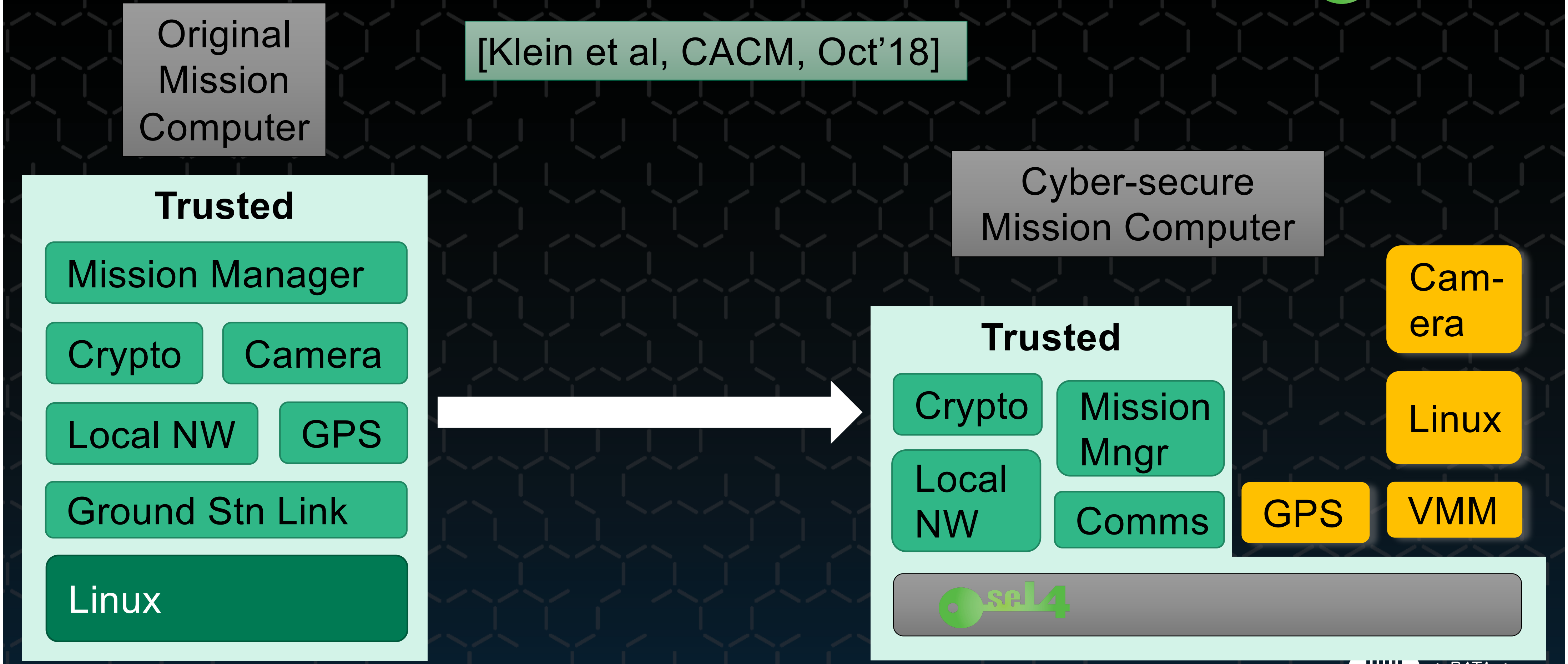
Linux

VMM

seL4

Incremental Cyber Retrofit

[Klein et al, CACM, Oct'18]



Original Mission Computer

Trusted

Mission Manager

Crypto

Camera

Local NW

GPS

Ground Stn Link

Linux

Cyber-secure Mission Computer

Trusted

Crypto

Mission Mngr

Local NW

Comms

GPS

Cam-era

Linux

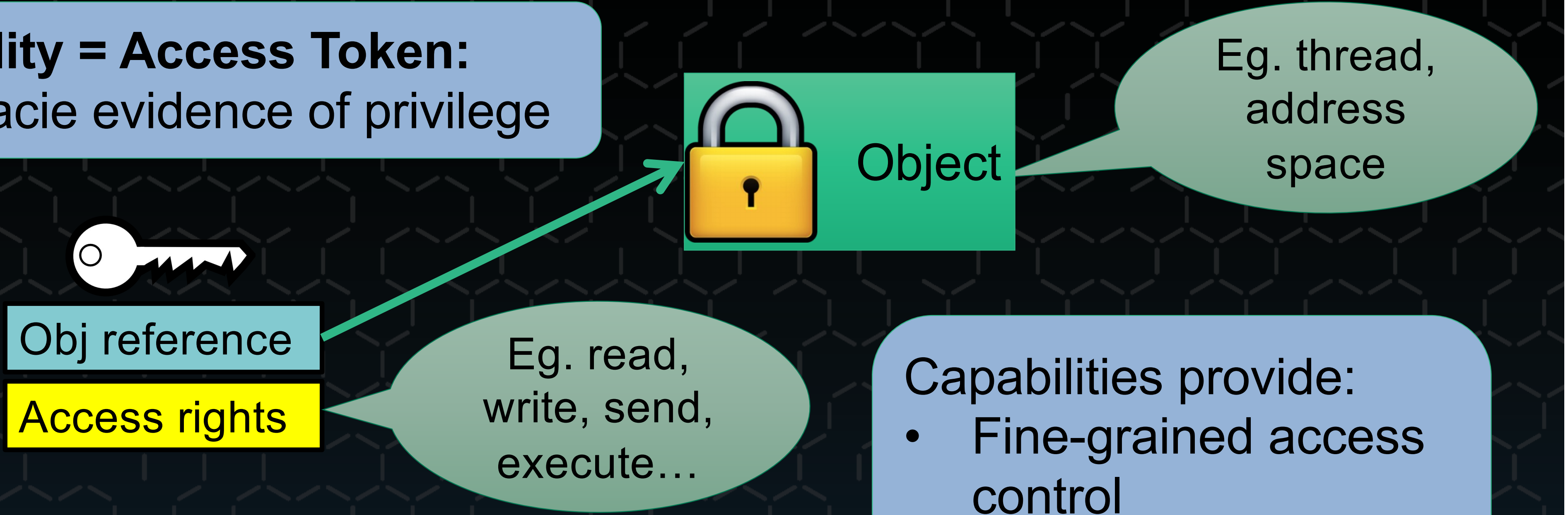
VMM

seL4

Security by Architecture

Core Security Mechanism: Capability

Capability = Access Token:
Prima-facie evidence of privilege



Capabilities provide:

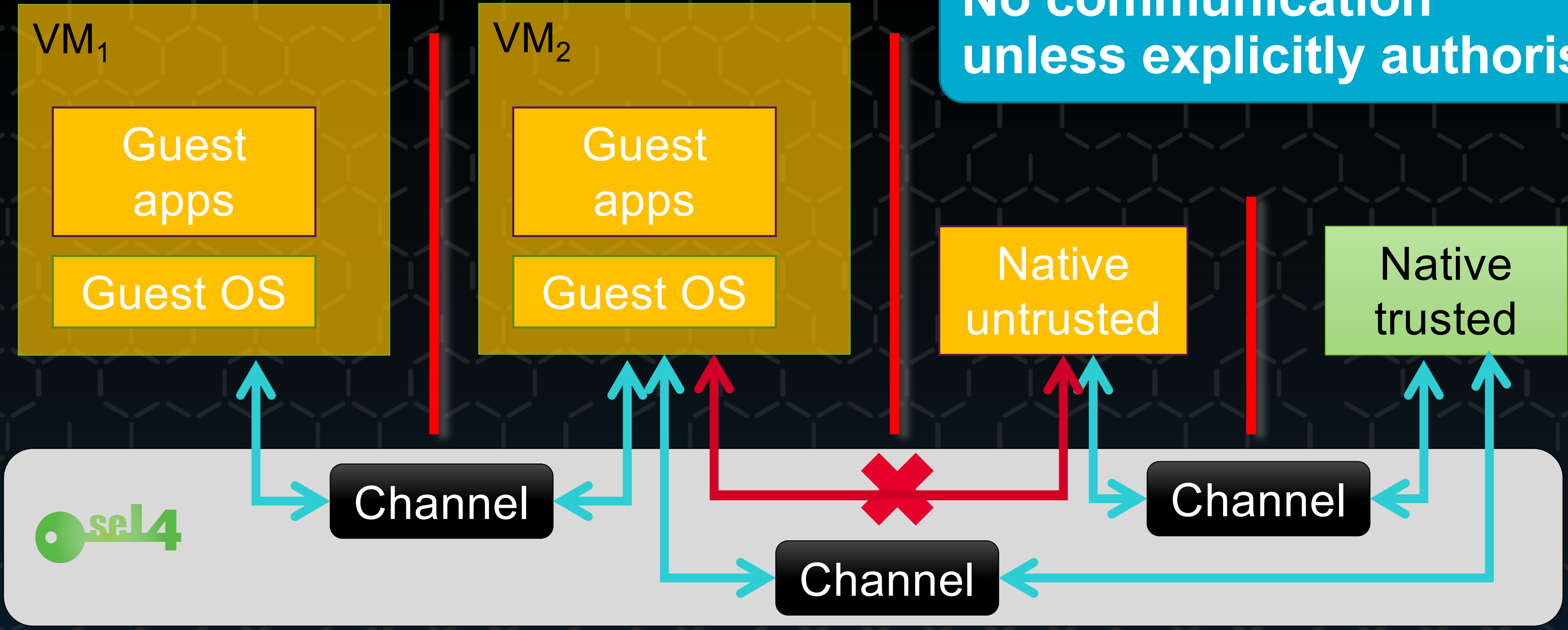
- Fine-grained access control
- Reasoning about information flow

Any system call is invoking a capability:

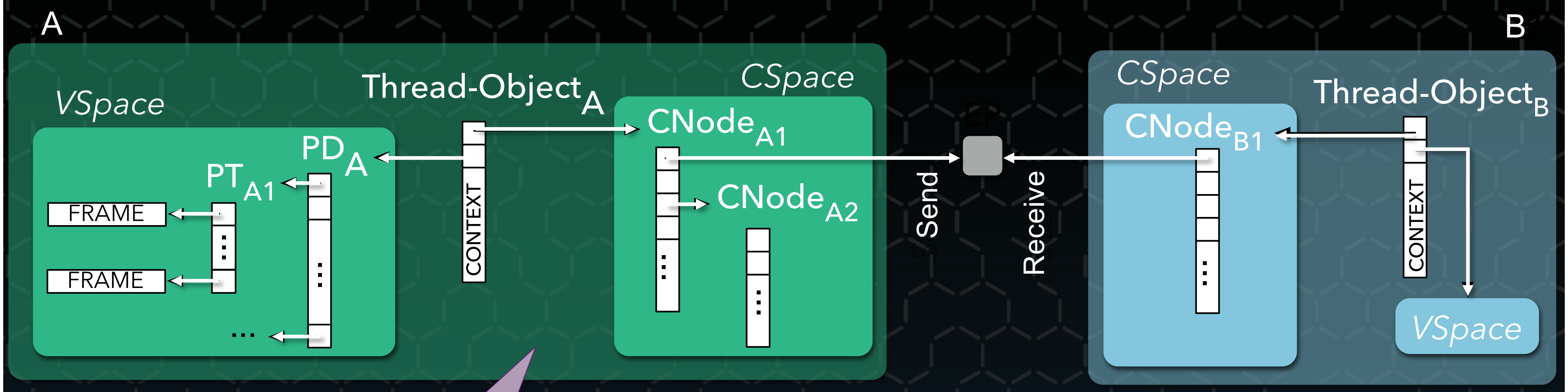
```
err = method( cap, args );
```

Controlled Communication via Caps

No communication unless explicitly authorised!

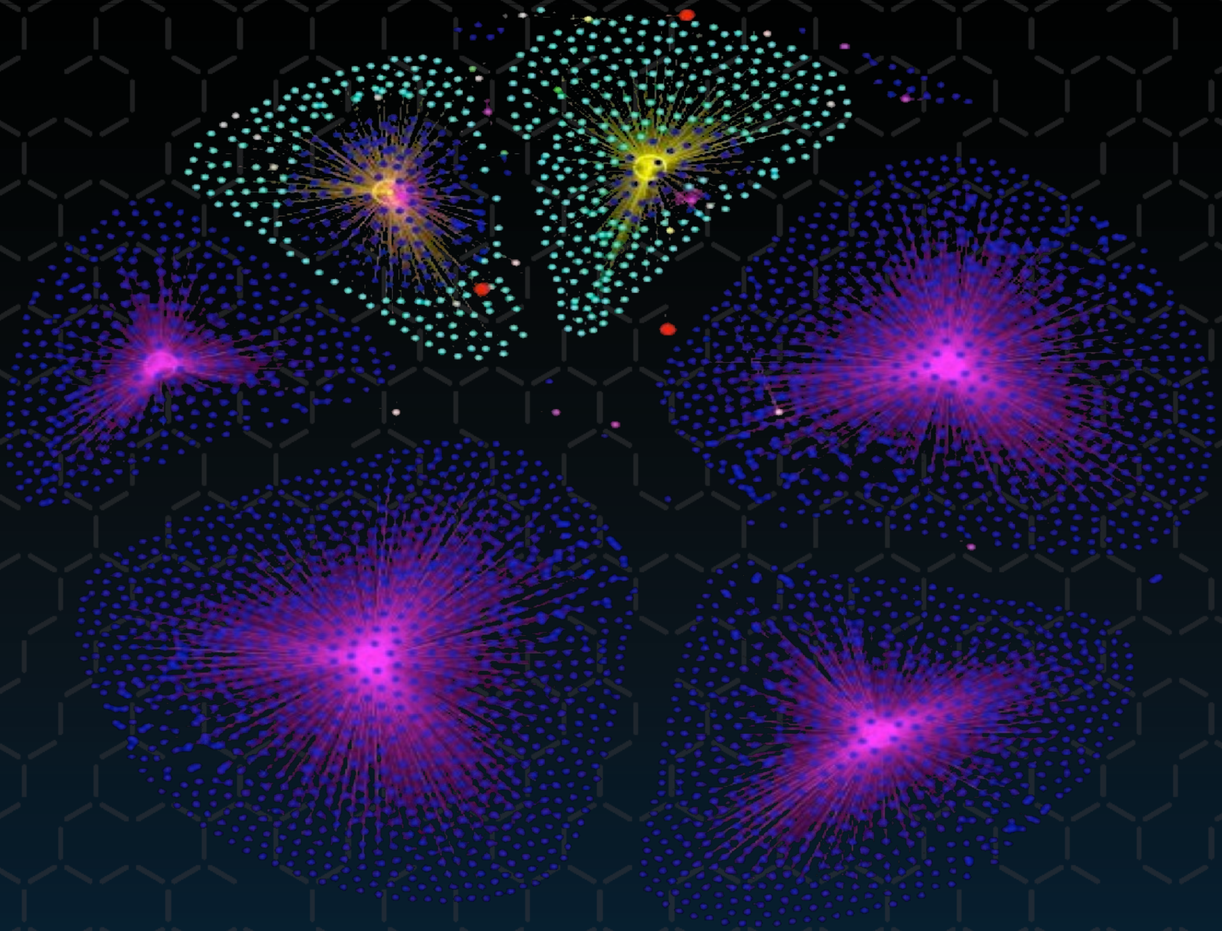


Issue: Capabilities are Low-Level



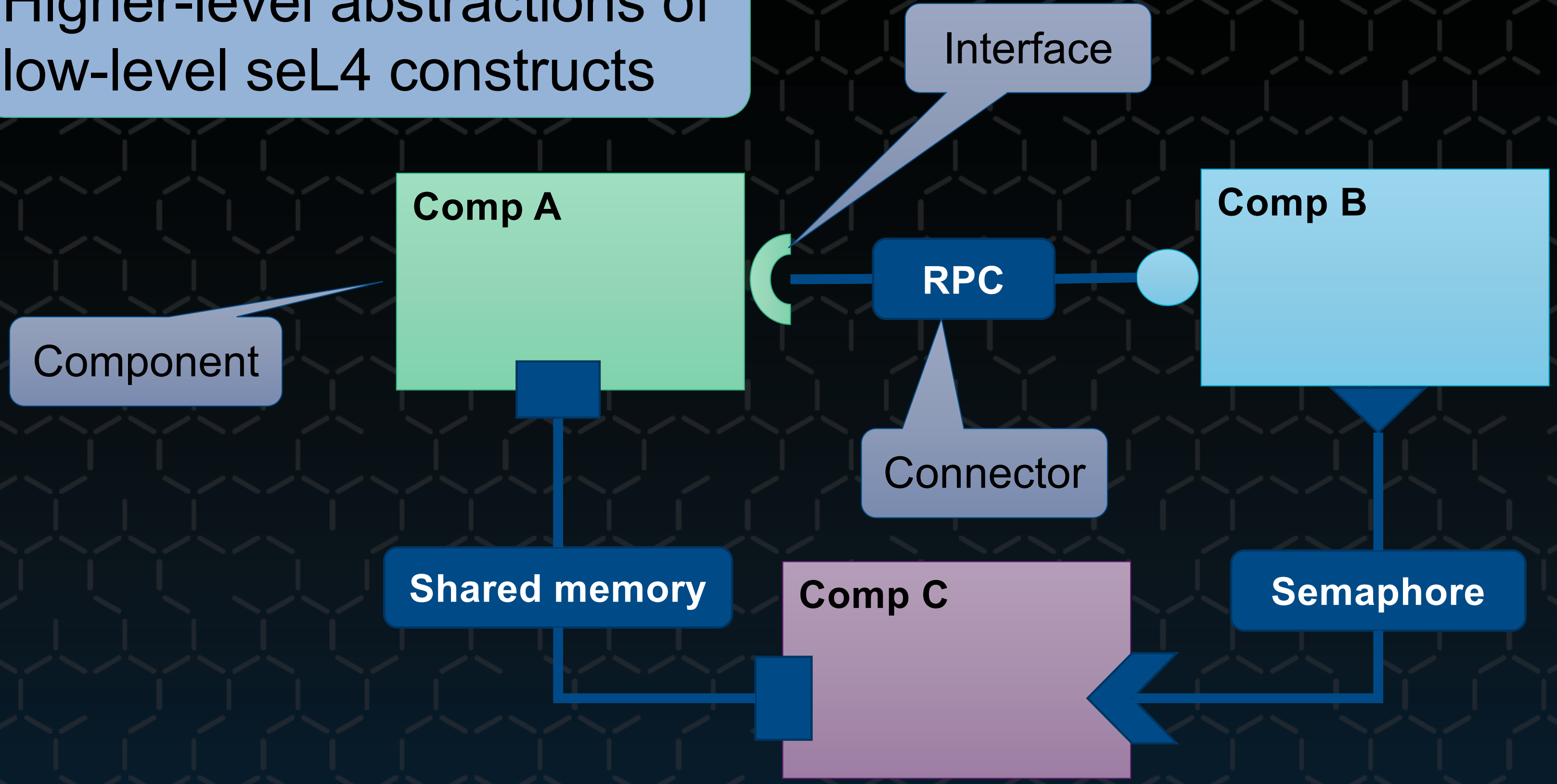
>50 capabilities
for trivial program!

Simple But Non-Trivial System

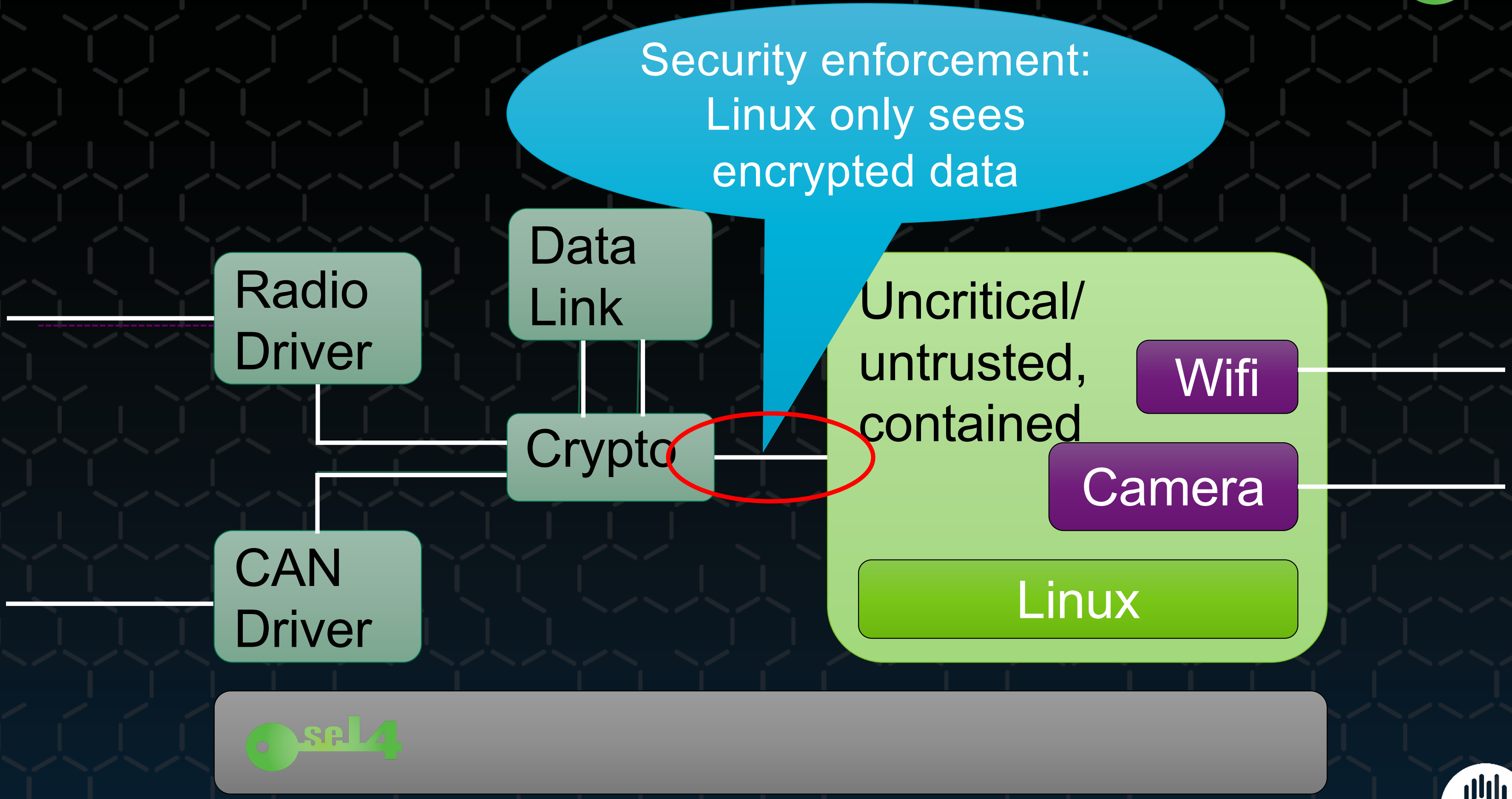


Component Middleware: CAmkES

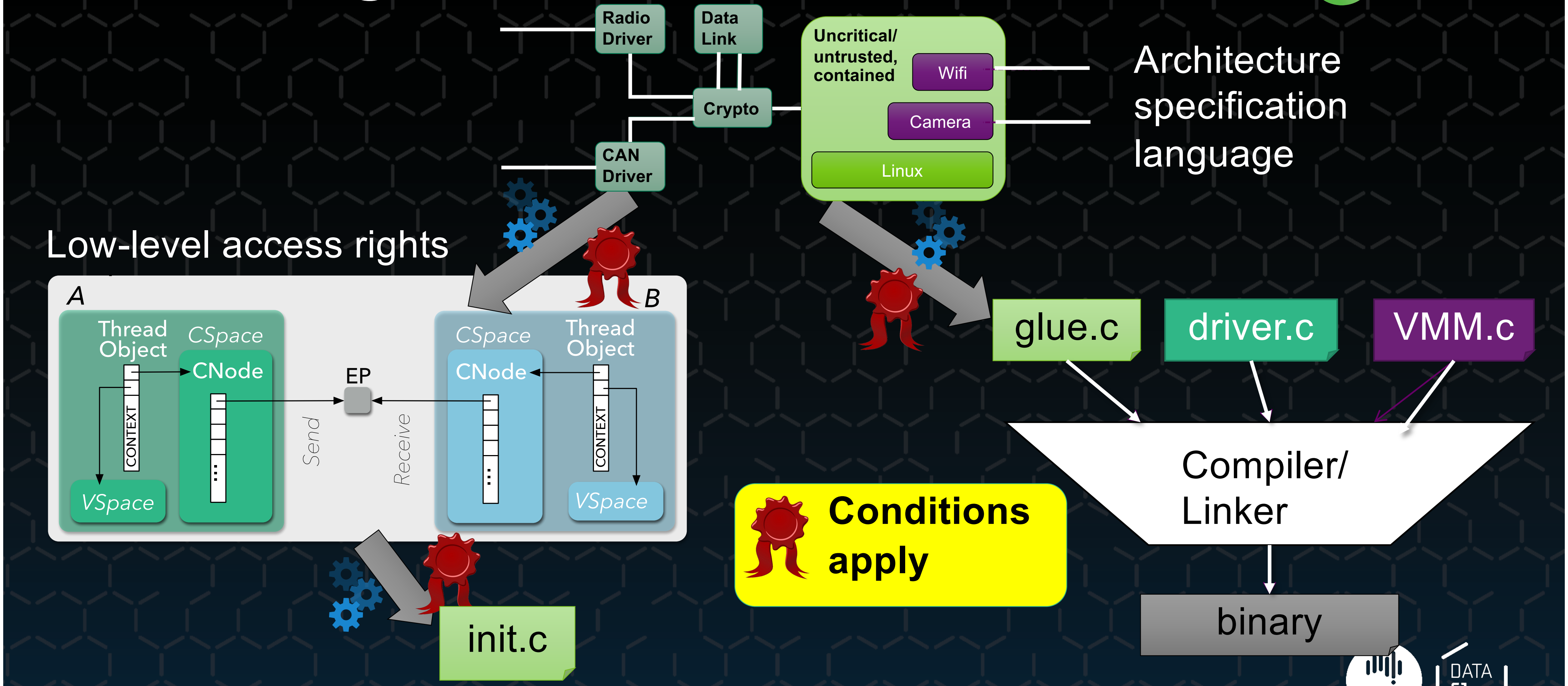
Higher-level abstractions of low-level seL4 constructs



HACMS UAV Architecture



Enforcing the Architecture

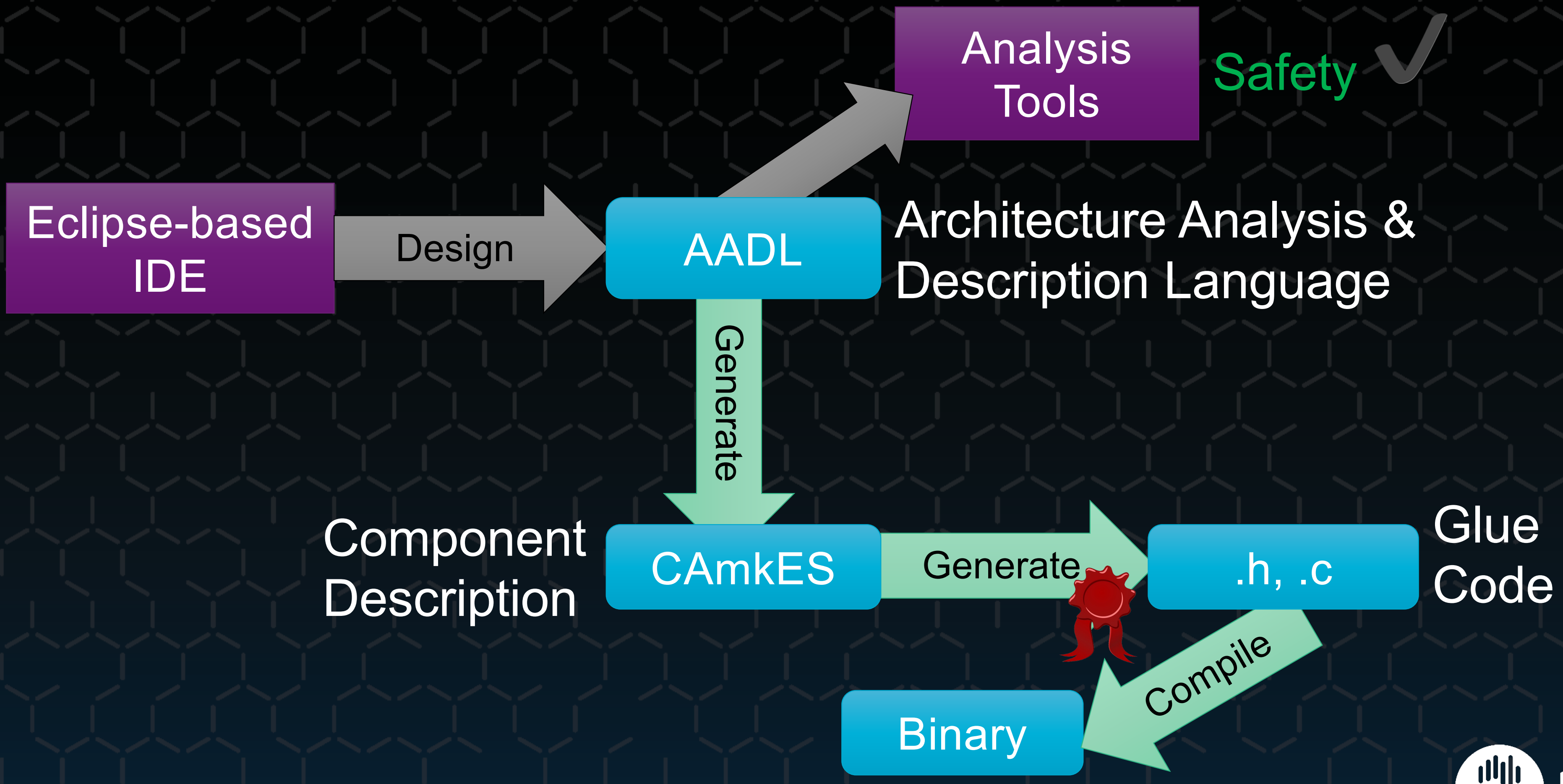


Architecture specification language

Low-level access rights

Conditions apply

Architecture Analysis



High Assurance Beyond the Kernel

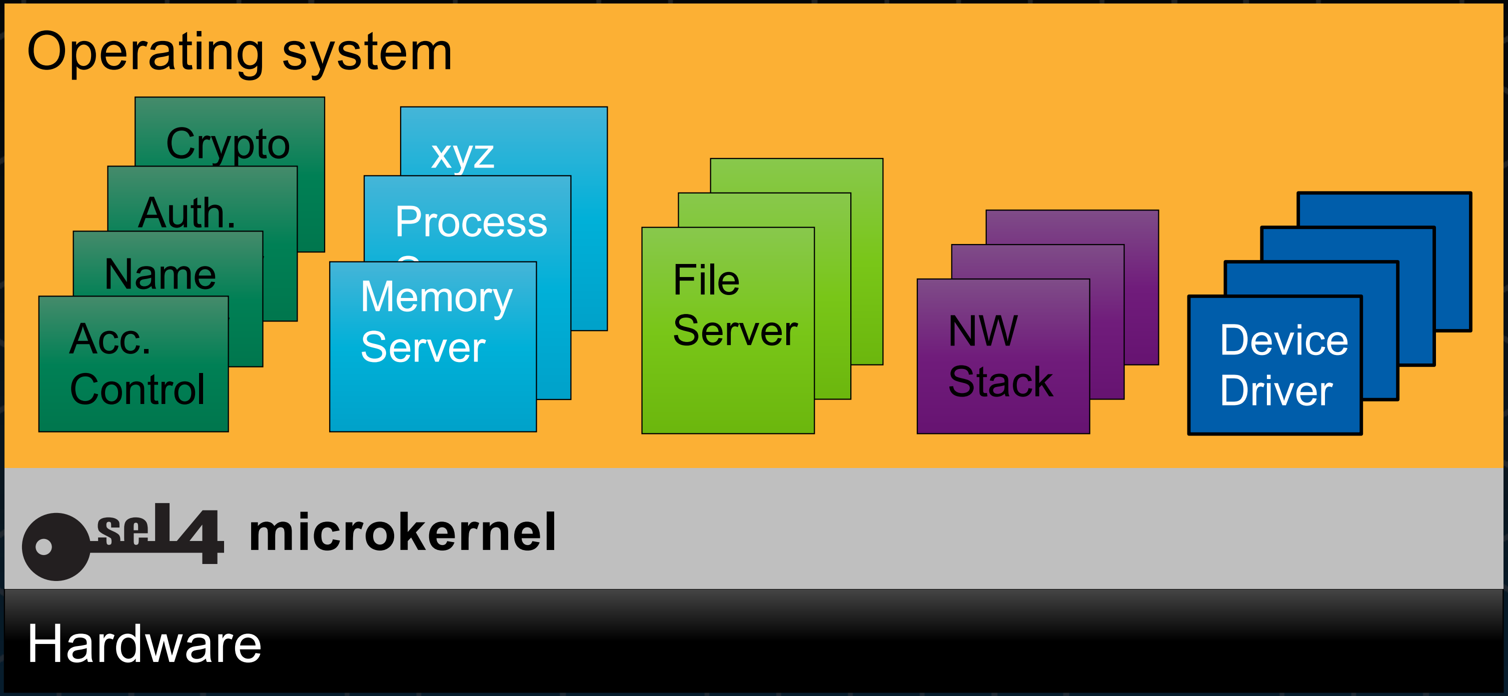
seL4

CSIRO

DATA 61

Microkernel ≪ TCB

OS structured in *isolated* components, minimal inter-component dependencies, *least privilege*



Operating system

Crypto

Auth.

Name

Acc. Control

xyz

Process

Memory Server

File Server

NW Stack

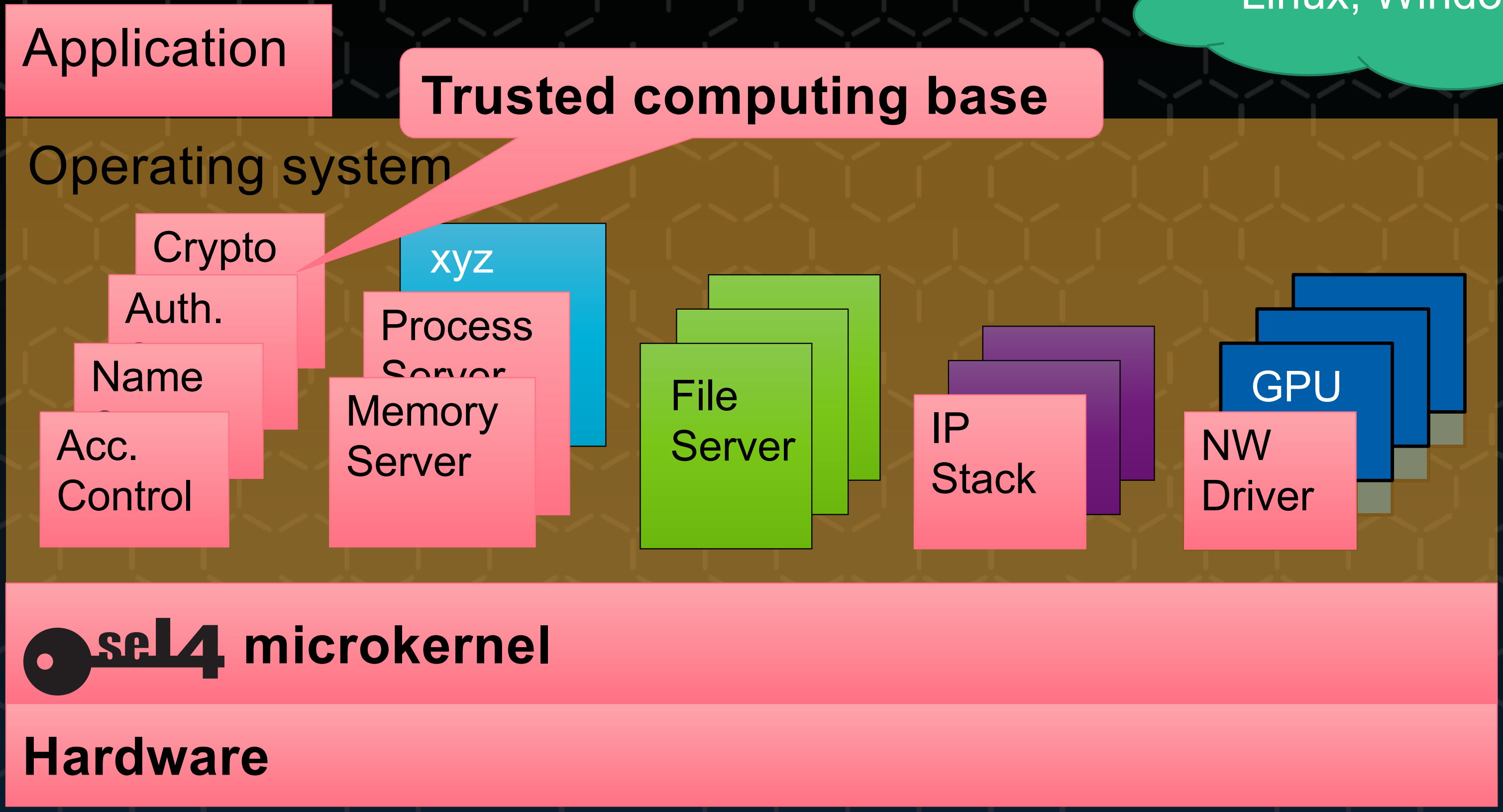
Device Driver

seL4 microkernel

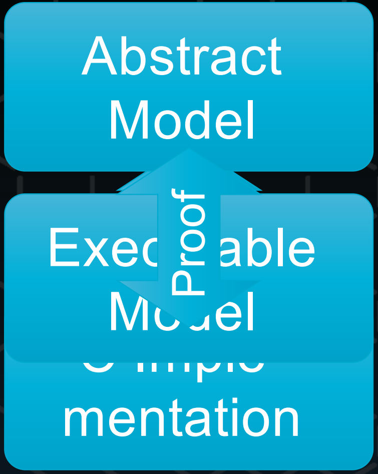
Hardware

Microkernel ≪ TCB

But *much* less than Linux, Windows…



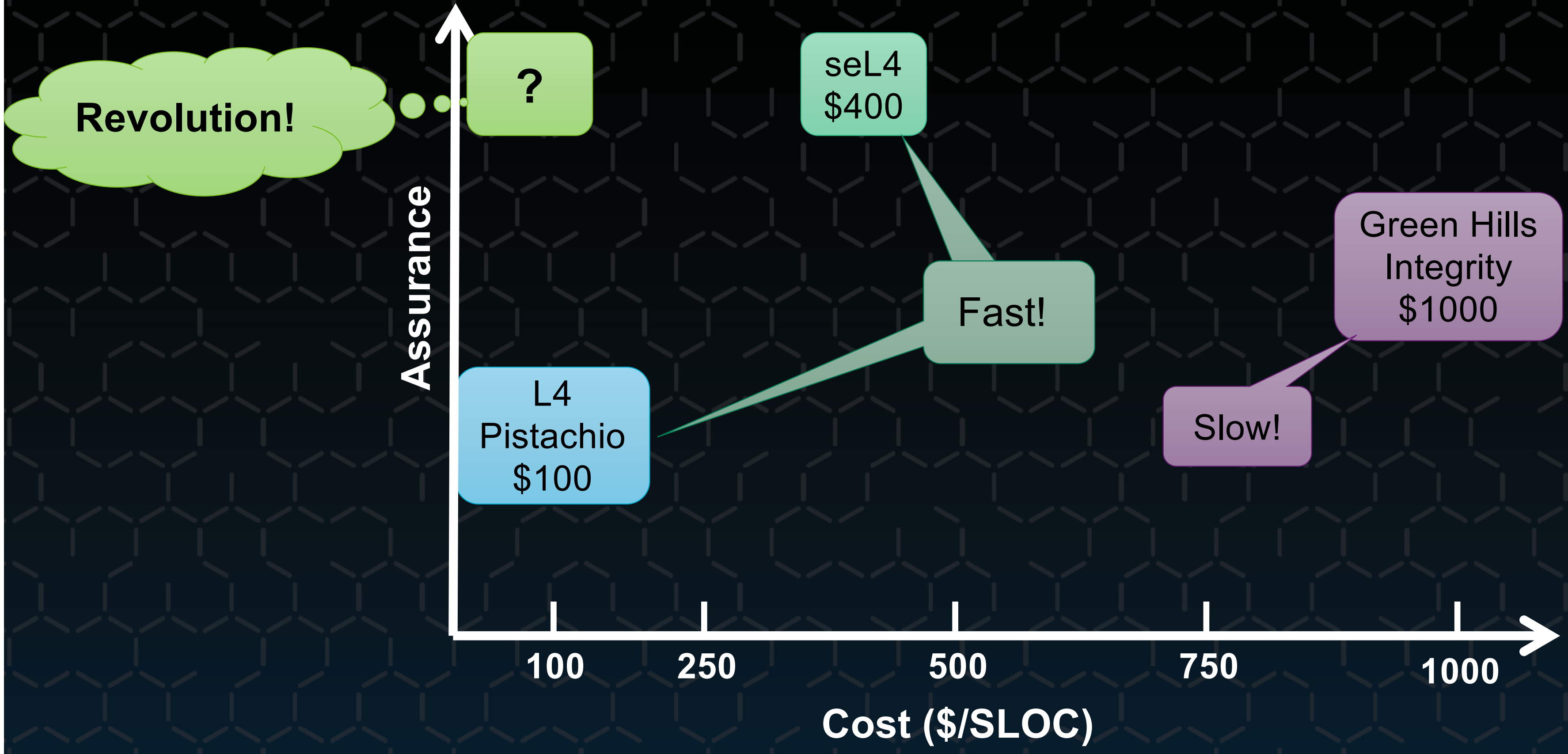
Verification Cost



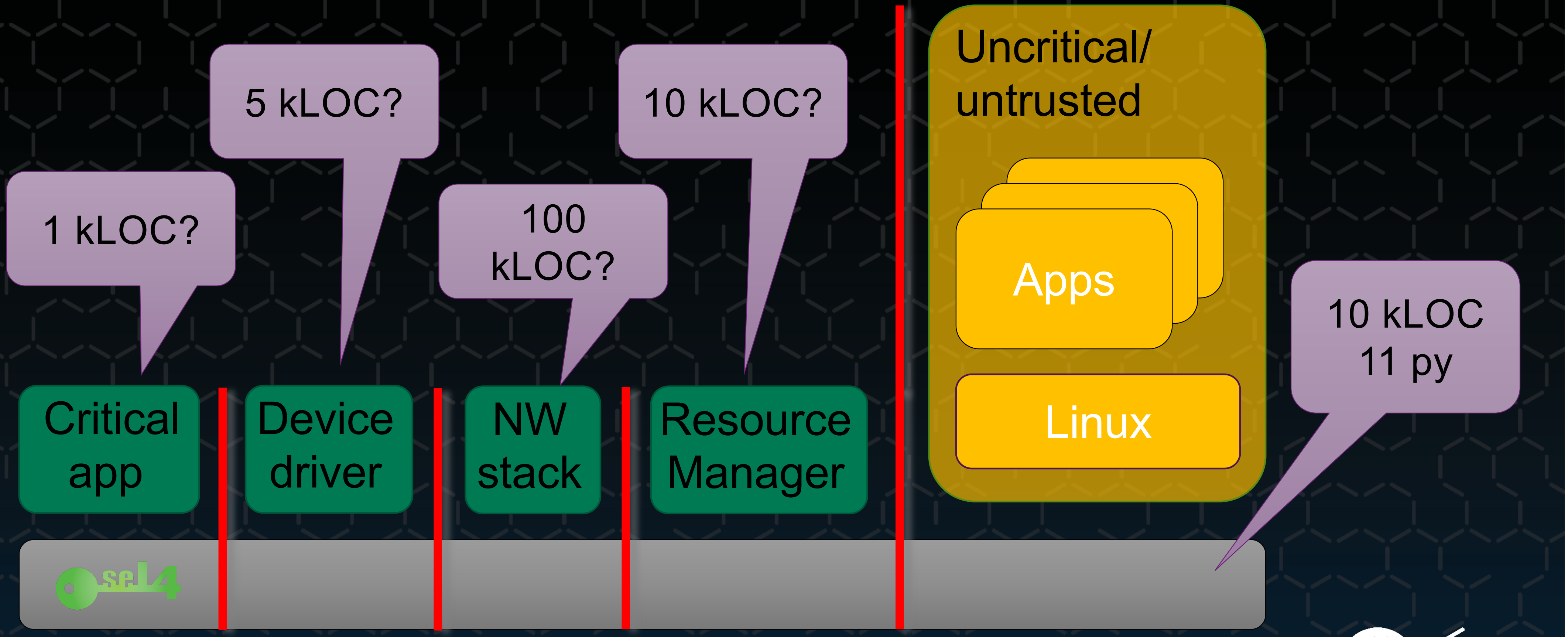
120,000 LoP, 8 py

50,000 LoP, 3 py

Life-Cycle Cost in Context



Beyond the Kernel

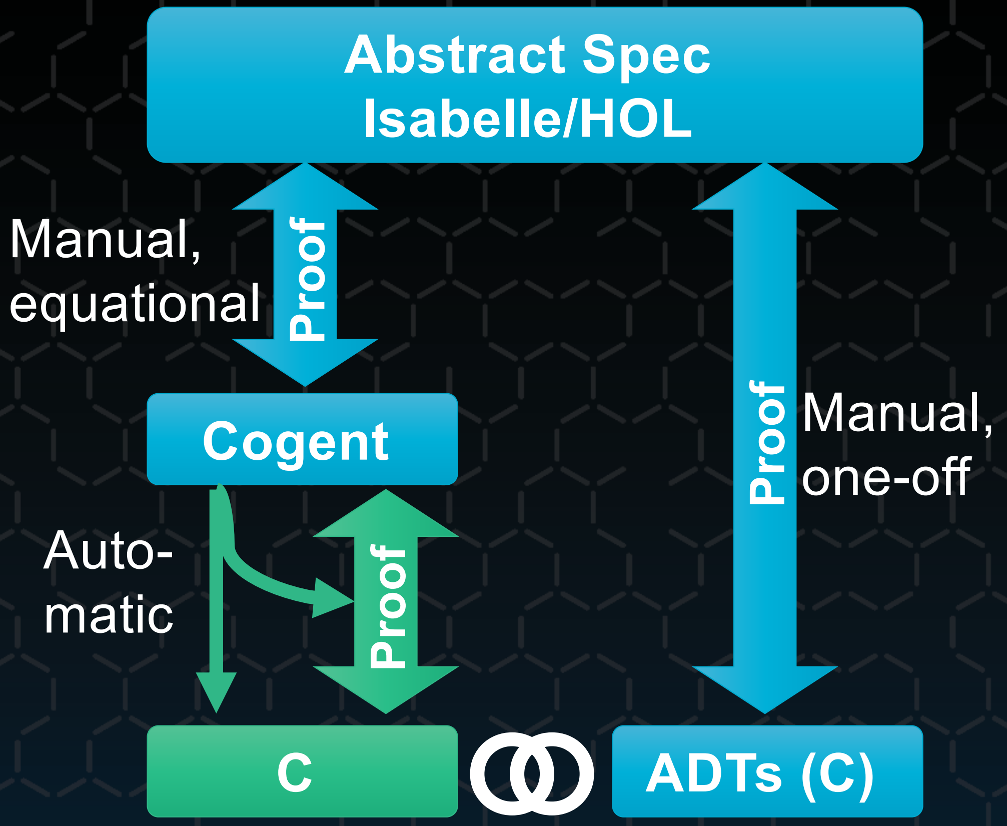


Cogent: Code & Proof Co-Generation

Aim: Reduce cost of verified *systems* code

- Restricted, purely functional *systems* language
- Type- and memory safe, not managed
- Turing incomplete
- File system case-studies: BilbyFs, ext2, F2FS, VFAT

[O'Connor et al, ICFP'16; Amani et al, ASPLOS'16]



Abstract Spec
Isabelle/HOL

Manual, equational

Proof

Cogent

Auto-matic

Proof

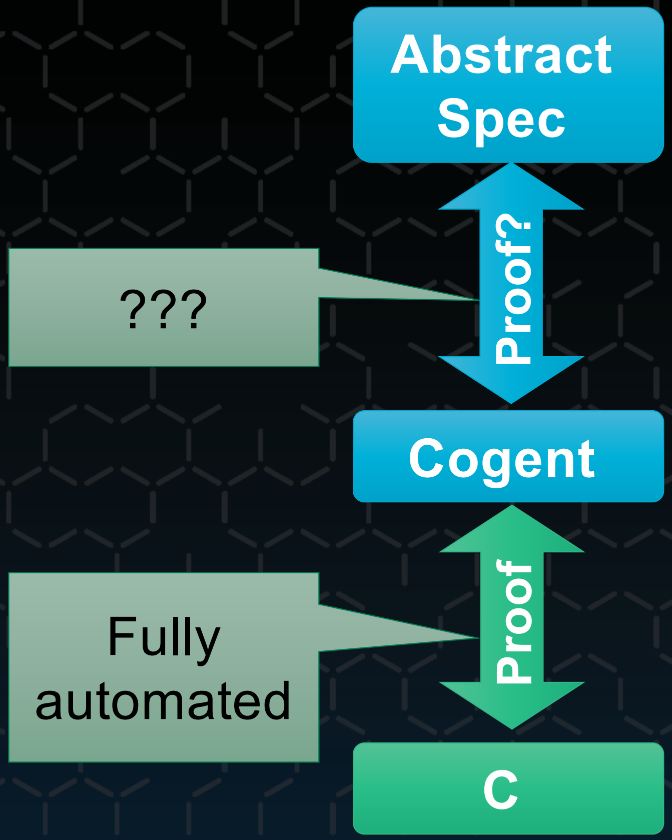
C

ADTs (C)

Proof

Manual, one-off

Dependable And Affordable?



Abstract Spec

Proof?

???

Cogent

Proof

Fully automated

C

Dependability-cost tradeoff:

- Reduced faults through safe language
- Property-based testing (QuickCheck)
- Model checking
- Full functional correctness proof

Spec reuse!

Work in progress:

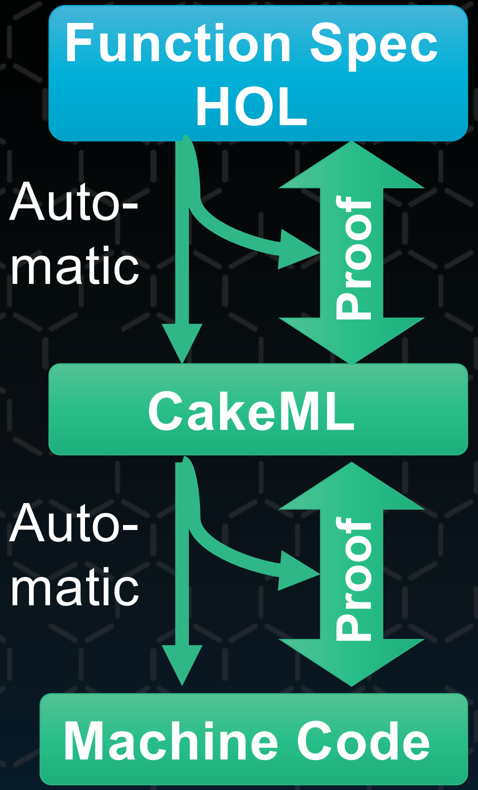
- Language expressiveness
- Reduce boiler-plate code
- Use for network stacks
- Use for device drivers

CakeML: Synthesising Code & Proofs

Aim: Reduce cost of verified *applications* code

- Impure, general-purpose functional language
- Type-safe, managed, garbage-collected, not memory-safe, Turing complete
- Verified run-time (GC etc)
- Compiles to binary for Armv6/8, x86, MIPS62, RISC-V
- Competitive performance

[Tan et al., ICFP'16]



CAmkES glue-code verification in progress

What Is Needed for Scaling Up

- More formal-methods experts
- Verified hardware and linking this to the operating system
- A Babel fish for formal methods
  - must overcome the composability problem (eg Cog–Isabelle)
- Better proof engineering tools and infrastructure, & more sharing!
  - seL4 alone now has ≈1M lines of proofs that need maintaining for evolving system
  - Problem becomes worse when dealing with whole system

Thank You!