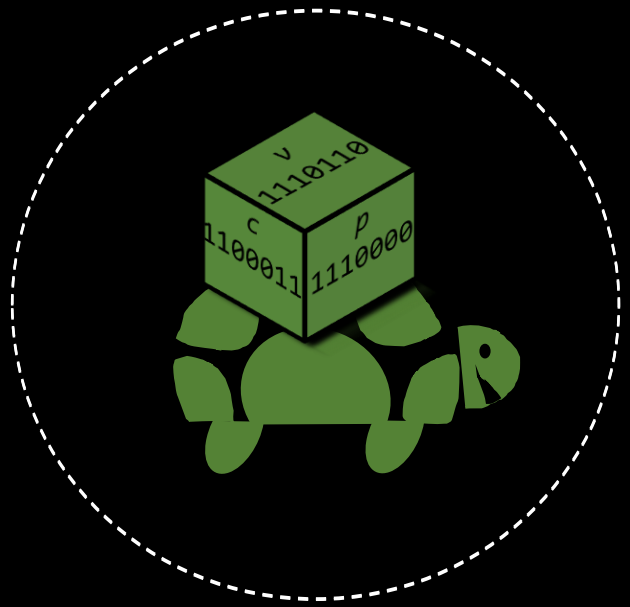
Freedom is the right of all sentient beings



überSpark:
Practical, Provable, End-to-End Guarantees on Commodity Heterogenous Interconnected Computing (CHIC) Platforms

Amit Vasudevan

(Software Engineering Institute/Carnegie Mellon University)

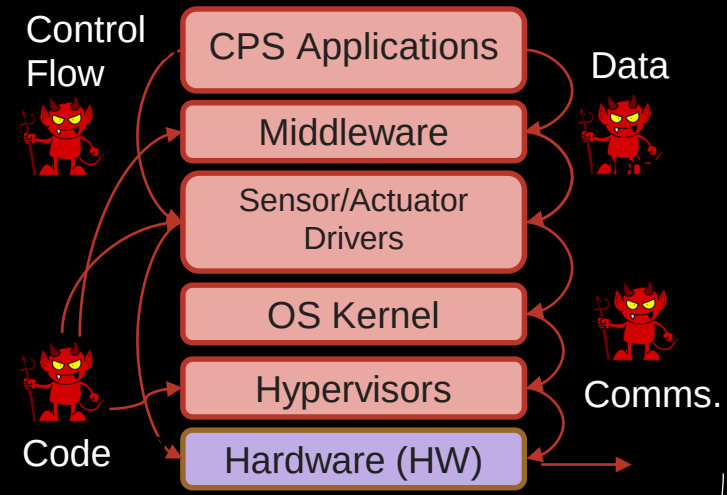
[Collaborators: Jeff Boleng (SEI/CMU), Anton Dimov Hristozov (SEI/CMU), Bruce Krogh (SEI/CMU), Raffaele Romagnoli (ECE/CMU), Ruben Martins (CSD/CMU), Atharv Saathe (ECE/CMU), Delbert Christman (Autonodyne LLC), Petros Maniatis (Google Research)]

https://uberspark.org | https://uberxmhf.org | https://hypcode.org

- This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

- DM21-0929

CHIC-centric Cyber (Physical) Systems



CHIC Stack Heterogeneity = Challenge: Platforms + Software Layers + Configuration & Interactions + Ownership + Development Pedigree

CHIC-centric CPS Implementations __demand__ fairly strong end-to-end assurance for security, control, timeliness, and correctness

Effective solution must meet these Goals: Provable + Cost-effective + Innocuous (cf. NASA)

"Neither impossible, nor impassable!"
-- Optimus Prime, TF

State-of-the-Art/Practice (SoA/SoP) and Shortcomings

CHIC Stack Heterogeneity = Challenge: Platforms + Software Layers + Configuration & Interactions + Ownership + Development Pedigree

SoA

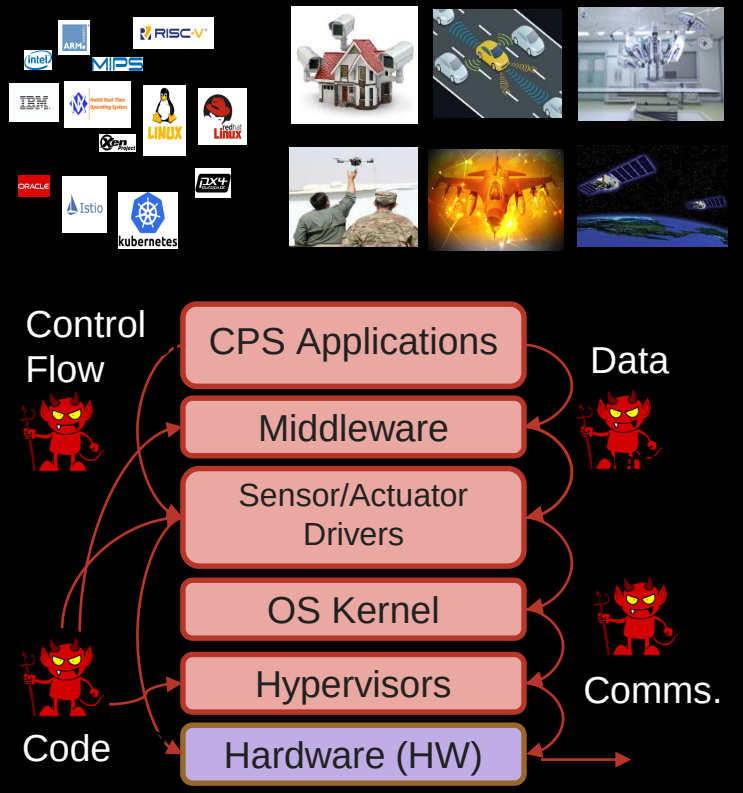
- Hardware/Software security kernels (e.g., micro-kernels, MILS, separation-kernels, hypervisors)
  - Not formally verified
  - Isolated components can still be exploited
  - No privileged disaggregation, kernel themselves are prone to vulnerabilities

SoA

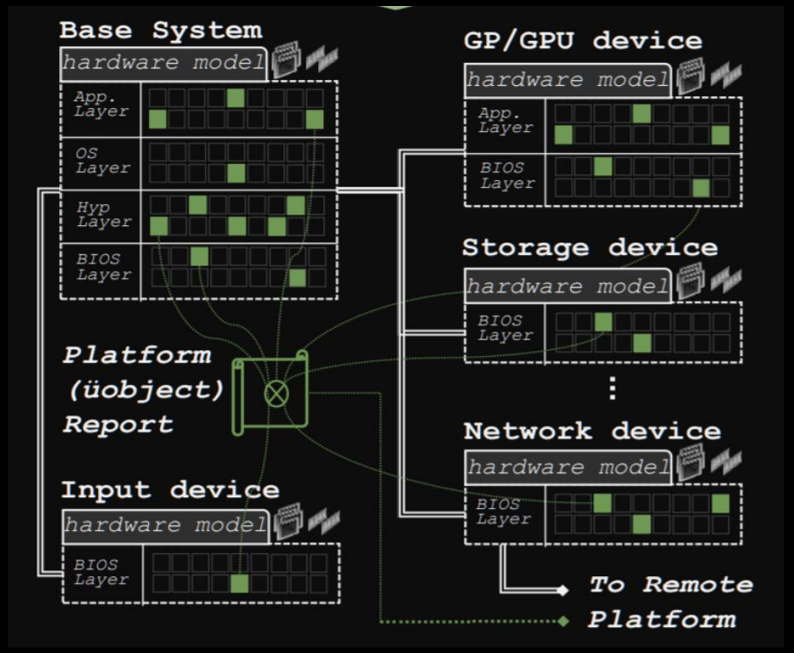
- Formally verified OS kernels (e.g., seL4, certiKOS, Ironclad. Verisoft etc.)
  - Focus is on verification methodology; written from scratch implementations
  - Treat CHIC-stack as monolith (e.g. run as VM) towards isolation property
  - Steep learning curve and (re-)verification cost for extensions and other properties
  - Minimal sensor/actuator device/driver support; verification methodology does not include devices/drivers (e.g., HACMS/seL4)

SoP

- Testing (functional, integration, security etc.)
  - Cannot provide complete coverage w.r.t a given functional or security requirement



überSpark Architecture Overview



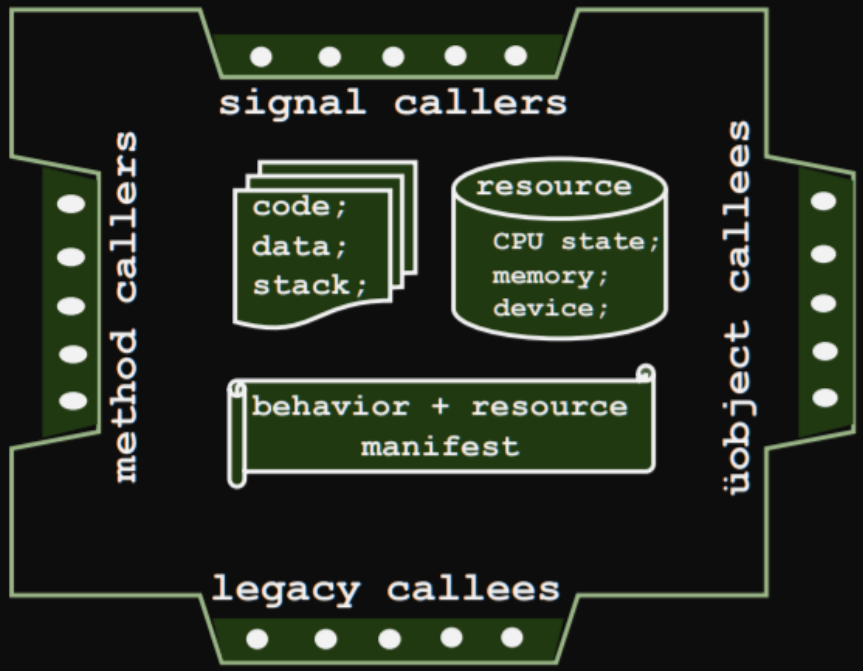
- **üobjects:** design time, singleton object abstraction for exclusive resource guards with secure interfaces
- **üobject collection:** runtime, protected group of üobjects with
  - *verified root-of-trust (vRoT)*
  - secure call routing
- **Assume-Guarantee (AG) reasoning on CHIC stack:** meshes unverified components and verified üobjects

- Flexible implementation on platform and CHIC-stack layer of choice
  - *app., kernel, driver subsystems*
- Retrofit at fine granularity
  - *containers, processes, portions of code*
- CHIC-AG reasoning allows incremental, composable verification in *developer friendly* manner
  - *Automated foundational properties (memory integrity, control-flow integrity, memory safety)*
  - *üobject specific properties (e.g., crypto)*
- Principled interfaces and resource closure allow state-of-the-art verification techniques on multi-threaded üobject executions

Provable + Cost-effective + Innocuous (cf. NASA)

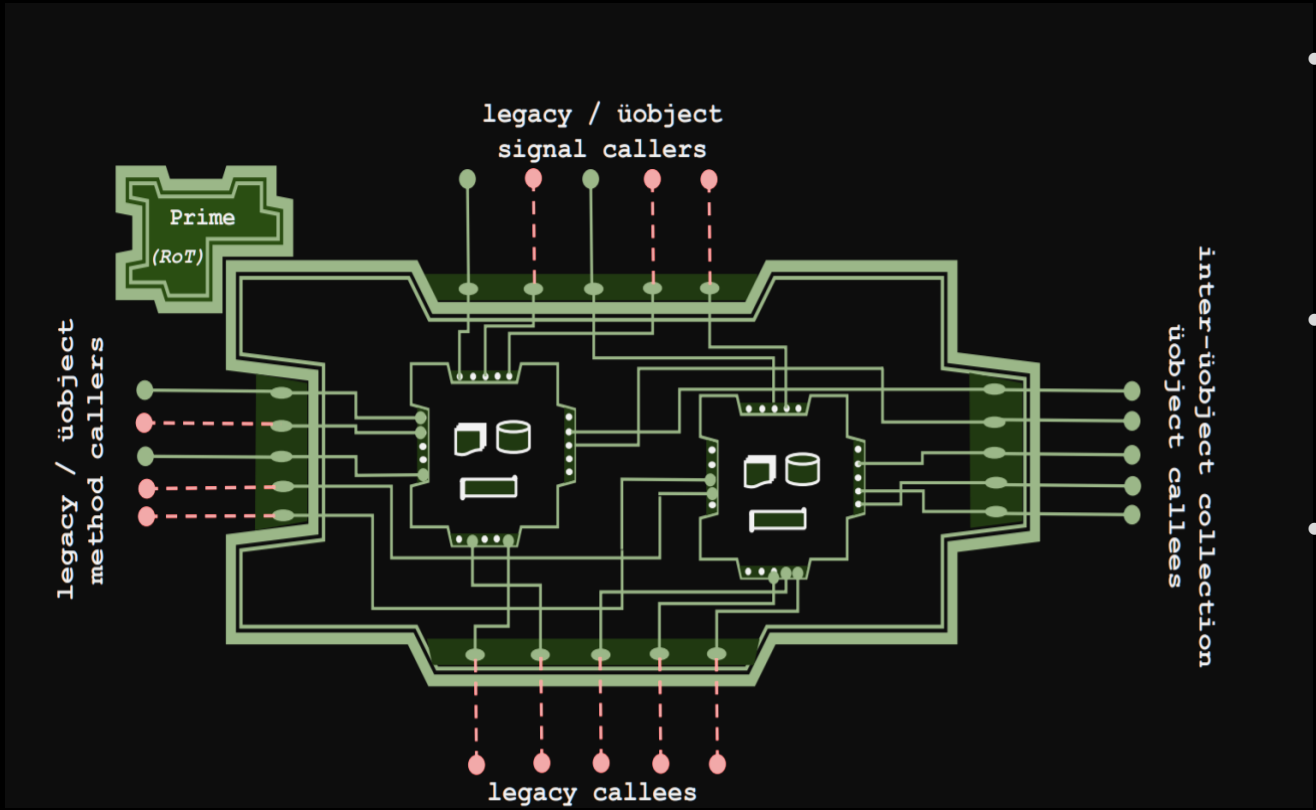
Amit Vasudevan, Petros Maniatis, and Ruben Martins: *"überSpark: Practical, Provable, End-to-End Guarantees on Commodity Heterogenous Interconnected Computing Platforms". ACM SIGOPS Operating Systems Review Journal Rev. 54-1, 2020*

üobject



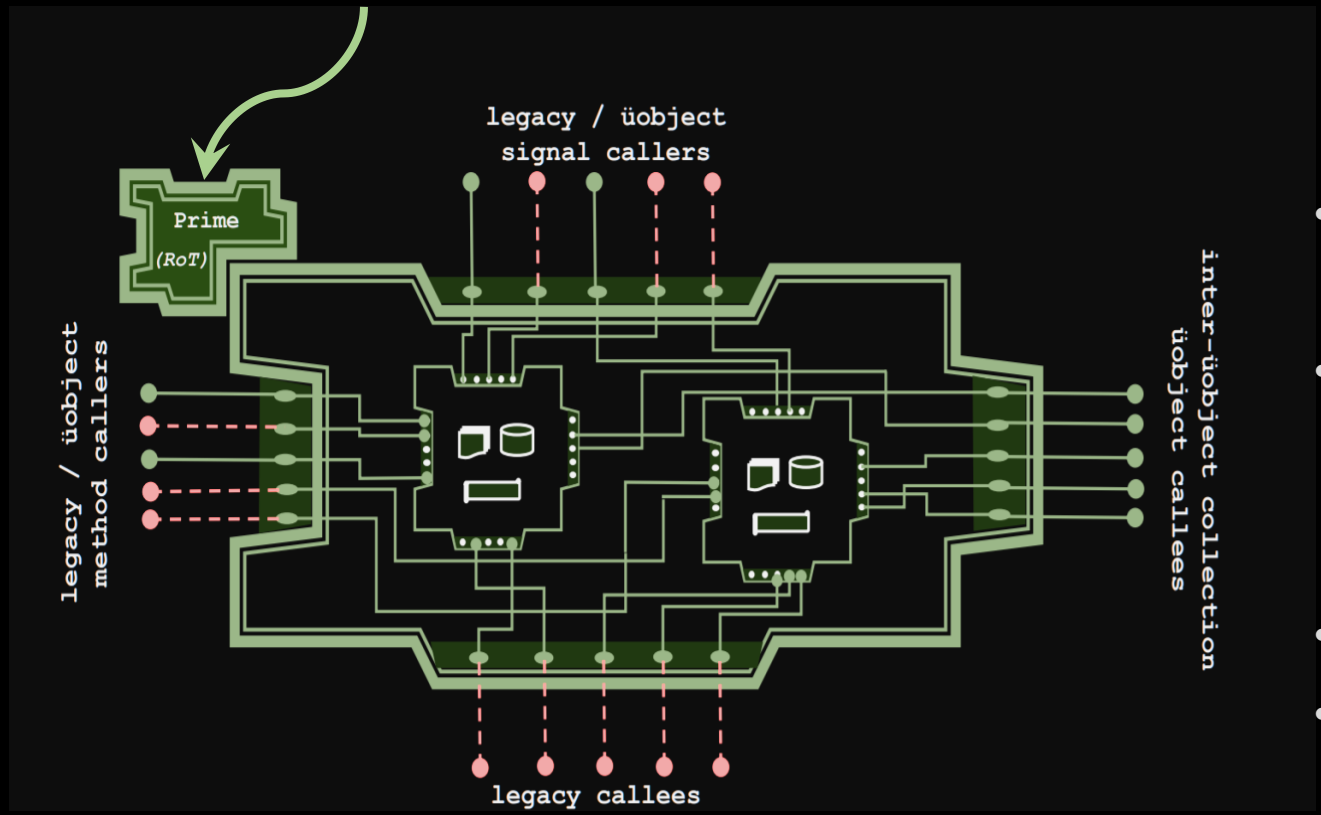
- Design time singleton object abstraction guarding exclusive indivisible system resources
- Principled entry, interruption, legacy code invocations and uobject invocations
  - Execution trace respecting program control flow enables use of state of the art program verification tools
  - Facilitate AG reasoning and composition
- Call-return interfacing
  - Handle various CHIC programming idioms
- Resource interface Confinement
  - Resource protection and access control
  - Support shared memory concurrency and linearizability → multithreaded execution and reasoning

üobject Collections



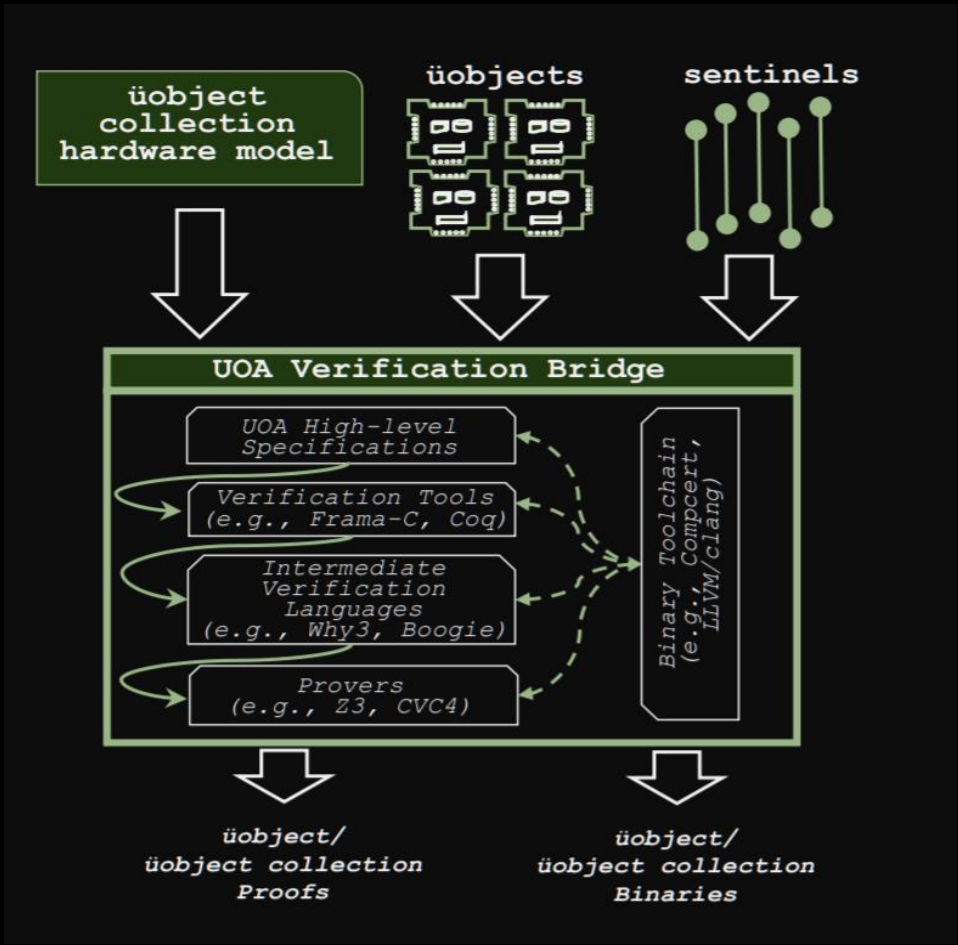
- Runtime abstractions that comprise a set of uobjects sharing a common memory space within a CHIC stack layer
- Bootstrapped by primes (coming up!) that form the root-of-trust entities
- Bridged via sentinel abstractions
  - Enforce call routings
  - Enable logical privilege separation
  - Uobject caller/callee mediation
  - Legacy component invocation
  - Both within and across uobject collections

Primes



- Special collection of üobjects responsible for boot-strapping üobject execution within a given collection in a protected manner
- Form Root-of-Trust entities
- Can employ different isolation mechanisms
  - Hardware assisted containerization
  - Software Fault Isolation (SFI)
- Setup üobject sentinels
- Initialize üobject collection CPUs, operating stacks and policies before kick-starting uobject interactions within the collection

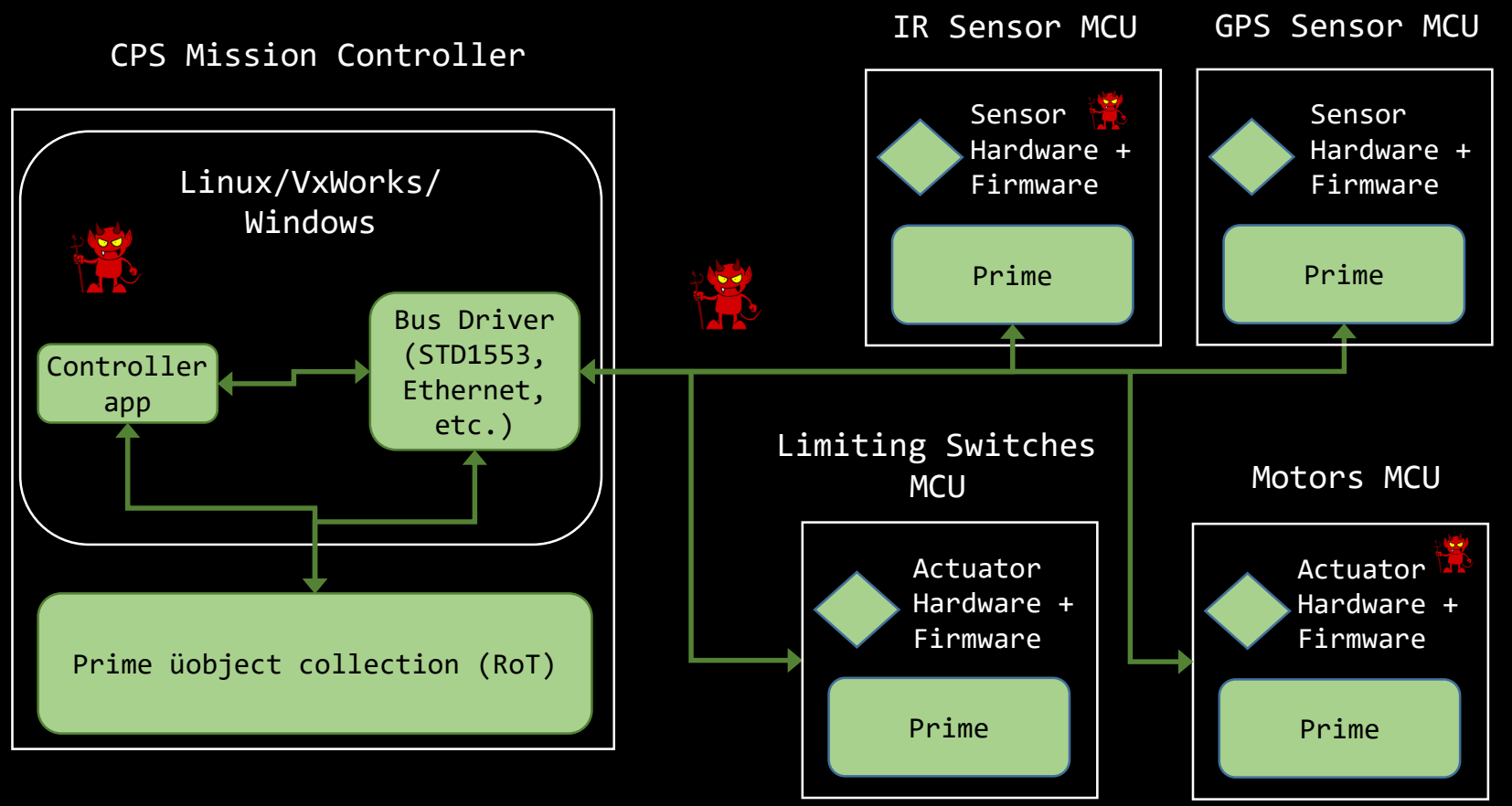
üobject Verification Bridge



- Facilitates Assume-Guarantee (AG) style reasoning on the CHIC stack
  - Üobject base invariants
  - Uobject specific properties
- Base invariants
  - Allow reasoning about uobjects in compositional manner
  - Designed to be verified automatically
- Invariants proven via combination of Proof Assumptions on Hardware (PAH) and Proof Obligations on Code (POC)
- Bridge high-level abstract language (BAL)
  - Abstracts Invariants, execution semantics and hardware model
  - Specifications corresponding to verification tool
  - Prove uobject specific properties via variety of verification tools

Amit Vasudevan, Sagar Chaki, Petros Maniatis, Limin Jia, Anupam Datta. “überSpark: Enforcing Verifiable Object Abstractions for Automated Compositional Security Analysis of a Hypervisor”. USENIX Security Symposium, 2016

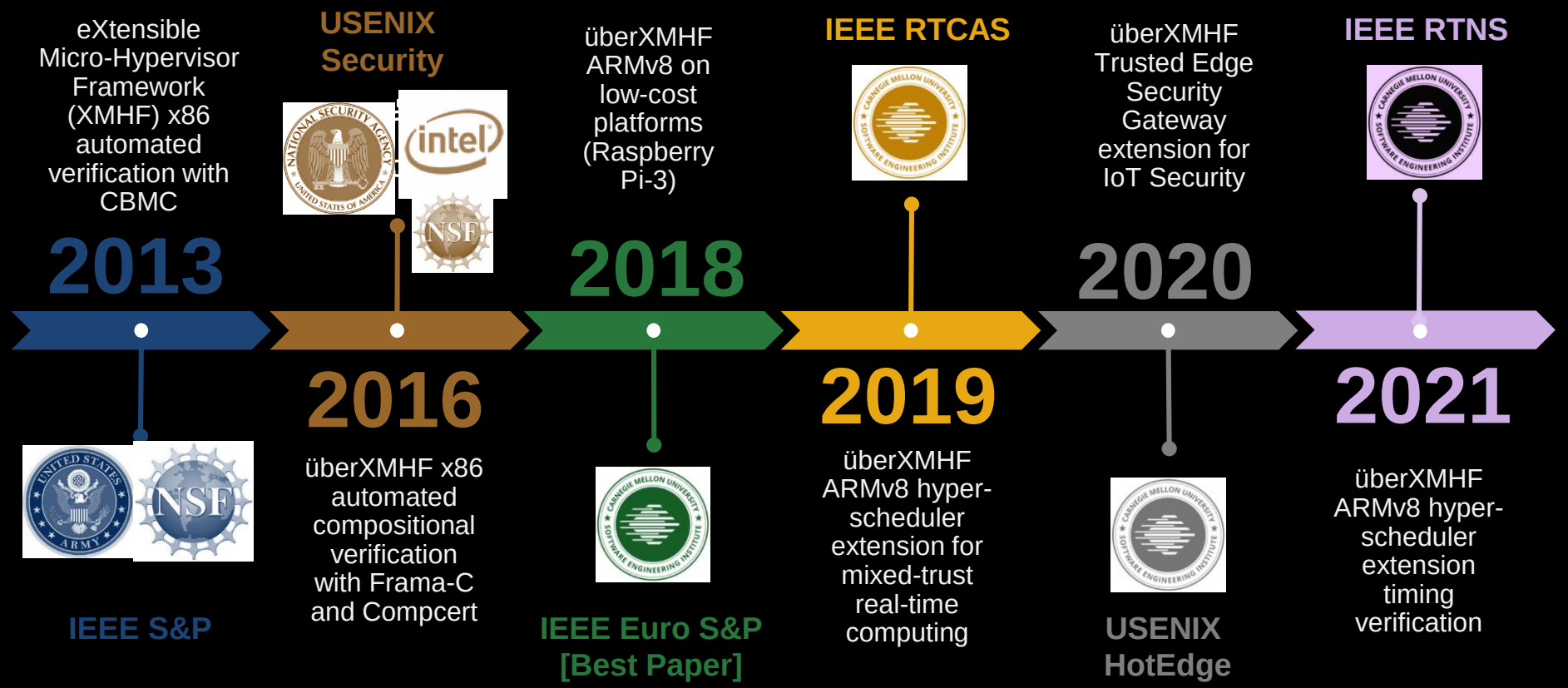
CHIC-centric CPS: Modular Provable End-to-End Guarantees



Foundational Steps: Verified RoT (Prime)

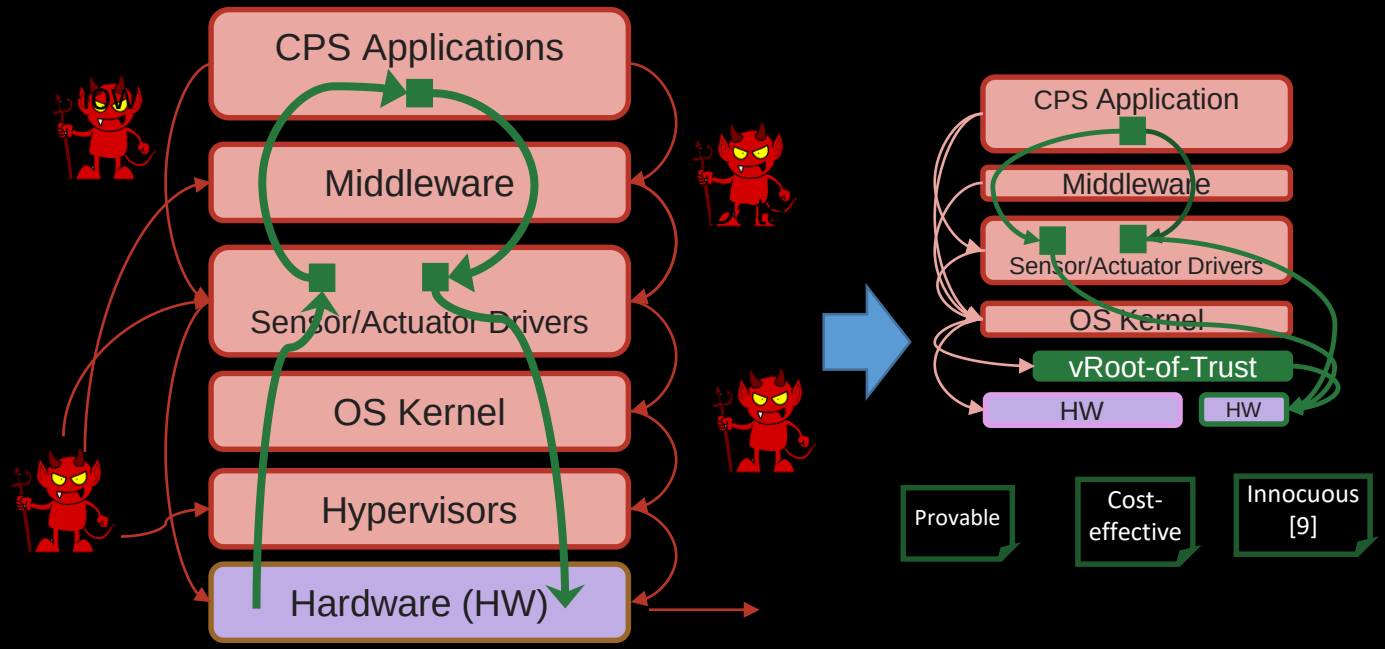
(Verified) Open-source Micro-Hypervisor Root-of-Trust: überXMHF (https://uberxmhf.org)

- Spatial Isolation
- Temporal Isolation
- Mediation
- Attestation



CHIC-centric CPS: Formally Verified Trusted Path

Integrity protection of the CPS application, sensor hardware/driver, root-of-trust logic with trusted path (control and data flow) between them



- CHIC-centric CPS Platform: Off-the-shelf (Amazon) Raspberry Pi3 Rover
- überXMHF micro-hypervisor vRoT
- üobjects realized on existing CPS application and Linux I2C drivers
- < 2% runtime overhead; 6 person weeks of effort
- Can complete mission even in presence of active attack!

DEMO: https://forums.uberspark.org/t/uapp-picar-s-demonstration-video-protecting-against-memory-attacks/281

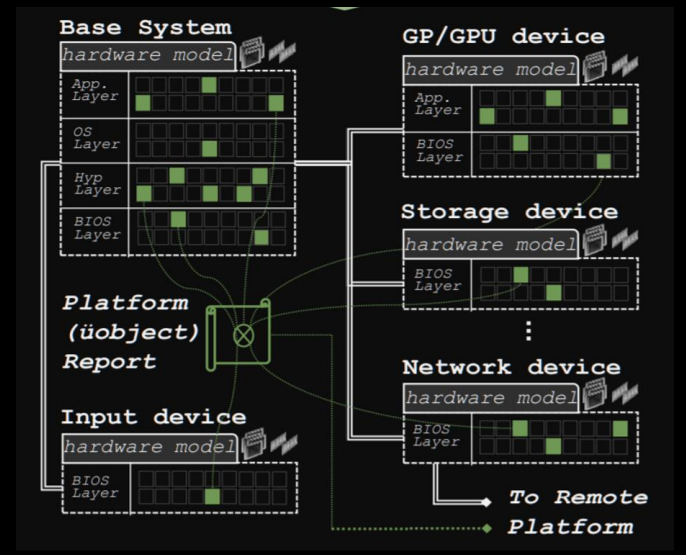
Anton Hristozov, Amit Vasudevan, Bruce Krogh, Raffaele Romagnoli, Atharv Saathe, Ethan Joseph, Ruben Martins: "A Low-cost, Highly-Customizable Security Platform for Robotics". *Submitted to the IEEE Intelligent Robot and Systems (IROS), 2021*

Present Activities and Roadmap

- üobject Programming Framework
  - Enforce üobject abstraction within any existing C & Assembly codebase for AG reasoning on CHIC-stack
  - Handle common programming idioms for apps, drivers and OS kernel
- Event-driven üobject executions and trusted-path scaling
- Modular, Secure, and Performant CPS Control Architectures
  - How should CPS missions control be architected/re-architected to get provable control, security and information-flow guarantees
- CHIC-AG reasoning system model and proof mechanization
  - Tri-fecta property for AG reasoning supporting interruptible executions across CHIC-stack layers (user, kernel, RoT)
  - Properties proven on TLA+ model → discharged automatically on code
  - Partial equivalence

überSpark: Summary

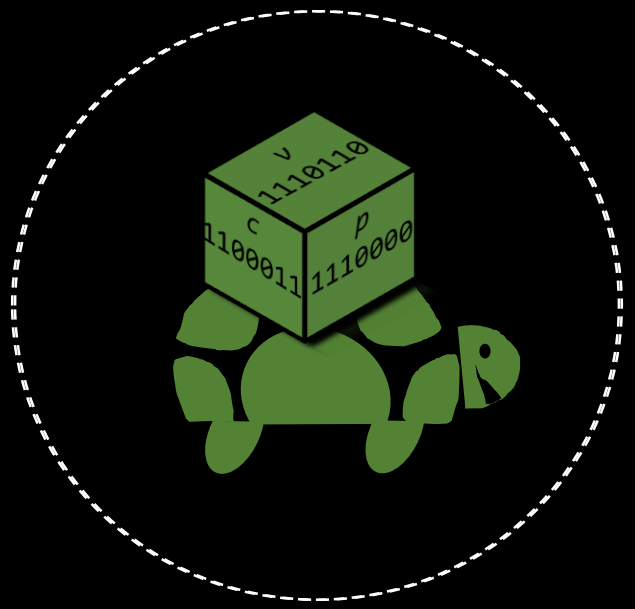
- Modular Provable Guarantees
  - Only focus on components and corresponding üobjects contributing to the guarantees
  - Adding a new üobject does not require re-verification of already verified üobjects
    - Might require additional meta reasoning (AG composition rules) at the verification bridge
- Developer Friendly programmatic abstraction and verification
- Commodity compatibility
  - *Existing* code on disparate platforms



Provable + Cost-effective + Innocuous (cf. NASA)

"Neither impossible, nor impassable!"
-- Optimus Prime, TF

Freedom is the right of all sentient beings



überSpark:
Practical, Provable, End-to-End Guarantees on Commodity Heterogenous Interconnected Computing (CHIC) Platforms

Amit Vasudevan

(Software Engineering Institute/Carnegie Mellon University)

Questions?

https://uberspark.org | https://uberxmhf.org | https://hypcode.org