

# Mixed-Trust Real-Time Computation

Dionisio de Niz ([dionisio@sei.cmu.edu](mailto:dionisio@sei.cmu.edu)), Technical Director, Assuring Cyber-Physical Systems Software Engineering Institute, Carnegie Mellon University

Certification authorities (e.g., FAA) allow the validation of different parts of a system with different degrees of rigor depending on their level of criticality. Formal methods have been recognized as important to verify safety-critical components. Unfortunately, a verified property can be easily compromised if the verified components are not protected from misbehaviors of the unverified ones (e.g., due to bugs). Thus, trust requires that both verification and protection of components are jointly considered.

A key challenge to building trust is the complexity of today's operating systems (OSs) making them impractical to verify. Building a trusted system is challenging because the underlying operating systems (OSs) that implement protection mechanisms are extremely hard (if even possible) to thoroughly verify. Thus, there has been a trend to minimize the trusted computing base (TCB) by developing small verified hypervisors (HVs) and microkernels, e.g., seL4, CertiKOS, and uberXMHF. In these systems, trusted and untrusted components co-exist on a single hardware platform but in a completely isolated and disjoint manner. We thus call this approach disjoint-trust computing. The fundamental limitation of disjoint-trust computing is that it does not allow the use of untrusted components in critical functionality whose safety must be assured through verification.

In this talk, we present the real-time mixed-trust computing (RT-MTC) framework. Unlike disjoint-trust computing, it gives the flexibility to use untrusted components even for CPS critical functionality. In this framework, untrusted components are monitored by verified components ensuring that the output of the untrusted components always lead to safe states (e.g., avoiding crashes). These monitoring components are known as logical enforcers. To ensure trust, these enforcers are protected by a verified micro-hypervisor. To preserve the timing guarantees of the system, RT-MTC uses temporal enforcers, which are small, self-contained codeblocks that perform a default safety action (e.g., hover in a quadrotor) if the untrusted component has not produced a correct output by a specified time. Temporal enforcers are contained within the verified micro-hypervisor. Our framework incorporates two schedulers: (i) a preemptive fixed-priority scheduler in the VM to run the untrusted components and (ii) a non-preemptive fixed-priority scheduler within the HV to run trusted components. To verify the timing correctness of safety-critical applications in our mixed-trust framework, we develop a new task model and schedulability analysis. We also present the design and implementation of a coordination protocol between the two schedulers to preserve the synchronization between the trusted and untrusted components while preventing dependencies that can compromise the trusted component.

Finally, we discuss the extension of this framework for trusted mode degradation. While a number of real-time modal models have been proposed, they fail to address the challenges presented here in at least two important respects. First, previous models consider mode transitions as simple task parameter changes without taking into account the computation required by the transition and the synchronization between the modes and the transition. Second, previous work does not address the challenges imposed by the need to preserve safety guarantees during the transition. Our work addresses these issues by extending the RT-MTC framework to include degradation modes and creating a schedulability model based on the digraph model that supports this extension.