Safety Assurance of Cyber-Physical Systems Through Secure and Verifiable Information Flow Control

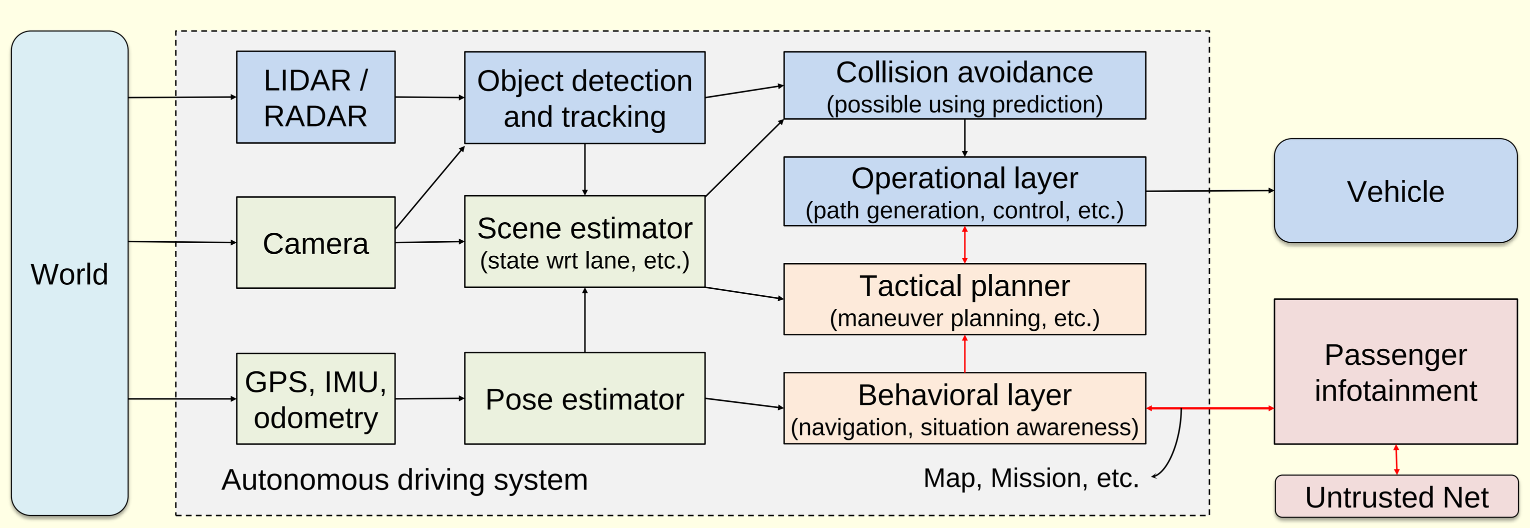
PIs: G. Edward Suh, Mark Campbell, Andrew C. Myers (Cornell University)

Objective

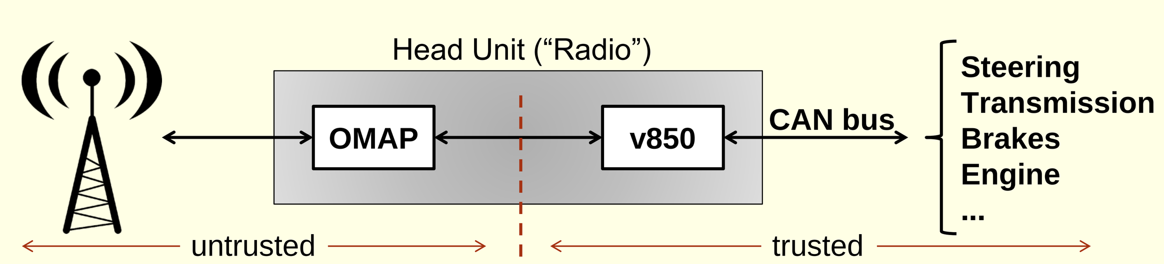
- **Problem**: safety-critical CPS is turning into complex networked systems ***vulnerable to remote attacks***
  - Internet connectivity + vulnerabilities in complex HW & SW
  - Implementation attacks: exploit bugs in HW or SW
  - Algorithmic attacks: tamper with inputs to control algorithms
- **Objective: *provable*** security assurance for safety-critical operations of autonomous driving systems
  - Focus on collision avoidance in self-driving cars
  - Formal assurance for security guarantees

Technical Approach

- **Co-design** hardware, software, and control algorithms
- Language-based Information Flow Control (IFC) for formal security assurance
  - Partition autonomous driving systems into multiple security levels
  - Build hardware and software with ***provable full-system information flow control (IFC)*** to ensure safety-critical operations can only be affected by untrusted inputs after an explicit endorsement
- Control algorithms to deal with untrusted information and provide quantitative safety assurance



Information Flow Control Example



- Real-world attack example on Jeep [Miller & Valasek 2015]
  - Head unit runs mainly on OMAP chip
  - OMAP communicates w/ v850 chip for remote door unlock, etc.
- **Vulnerability:** software updates including v850 software are unsigned & performed via head unit
- **IFC solution:** ensures integrity of software updates (e.g., explicit endorsement after verifying signatures)

Verifiably Secure Hardware

- Today's hardware is insufficient to protect safety-critical CPS platforms
  - No capability for fine-grained IFC across heterogeneous modules
  - No protection against timing interference
  - No formal security guarantee
- Redesign architecture for comprehensive and verifiable "Integrity" protection assurance
- **Formal assurance:** security type system for Verilog
  - Associate security labels with hardware signals
  - Statically check hardware-level information flows

```
reg  [18:0] {L} tag0[256];
reg  [18:0] {H} tag1[256];
wire [7:0]  {L} index;

// Par(0) = L  Par(1) = H
wire        {Par(way)} way;
wire [18:0] {Par(way)} tag_in;
wire        {Par(way)} write_enable;
```

```
always @ (posedge clock) begin
  if (write_enable) begin
    case(way)
      0: tag0[index] = tag_in;
      1: tag1[index] = tag_in;
    endcase
  end
end
```

Security check

Security check in the type system guarantees:

- No explicit information flow from **H** to **L**
- No unintended timing channels: when the label of an instruction is **L**, its execution time should only be affected by **L** hardware state

SW-Level Information Flow Control

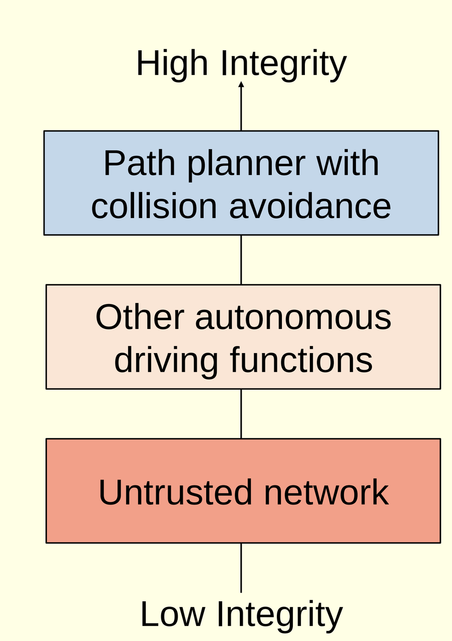
- Information-flow type systems enforce strong security properties assuming trustworthy hardware
  - Noninterference: No information flow from untrusted source to trusted sinks
  - Robust endorsement: trusted data influenced by untrusted data in circumscribed ways

```
MapData{U}  map;
Location{T} destination;
Route{U}    naviplan;
Path{T}     pathplan;

// compute the navigation route using map
naviplan.genRoute(map, destination);

Waypoint{U} w = naviplan.nextWaypoint();

// check and endorse next step to high
// integrity if it checks out vs. sensor data
endorse(w, L to H) if (verifiedStep(w)) {
    // generate a trusted vehicle path
    pathplan.genPath(w, ...);
}
```



- Extend language-based information flow control to *handle **integrity** and **availability** on **modern SoCs***
  - Prove ***the use of correct information flows***
  - Handle information flows through heterogeneous element

⇒ Full stack of hardware + software satisfies strong information flow security properties

Control Algorithms and Safety Analysis

- *Develop collision avoidance algorithms* to handle untrusted inputs such as detailed maps, traffic info, etc.
  - Strategy (1): sensor verification of map; preload key known landmarks; verify landmarks while driving
  - Strategy (2): verification of plan via sensors; develop plan with untrusted map; build occupancy map via sensor data only in front of car; verify plan will not cause collision
  - Strategy (3): contingency planning; develop nominal plan with untrusted map; develop a family of plans based on the potential of untrusted data; optimize plan switching logic to provide collision guarantees; utilize multiple sensors
- Probabilistic collision analysis of the integrated system
  - Quantitative analysis of the safety–collision probability
  - Investigate the tradeoff between collision probability and security protection levels (timing guarantees, amount of information, size of TCB, etc.)

Segway Autonomous Driving Testbed

- Integrates all three components: HW, SW, control
  - **Segway robot** with cameras, lidar, and IMU/GPS. Use for year-round testing in controlled environments.
  - FPGA-based hardware platform: ARM + custom RISC-V processor
  - Software in Jif programming language
  - Migrate to the Skynet autonomous driving vehicle in the future