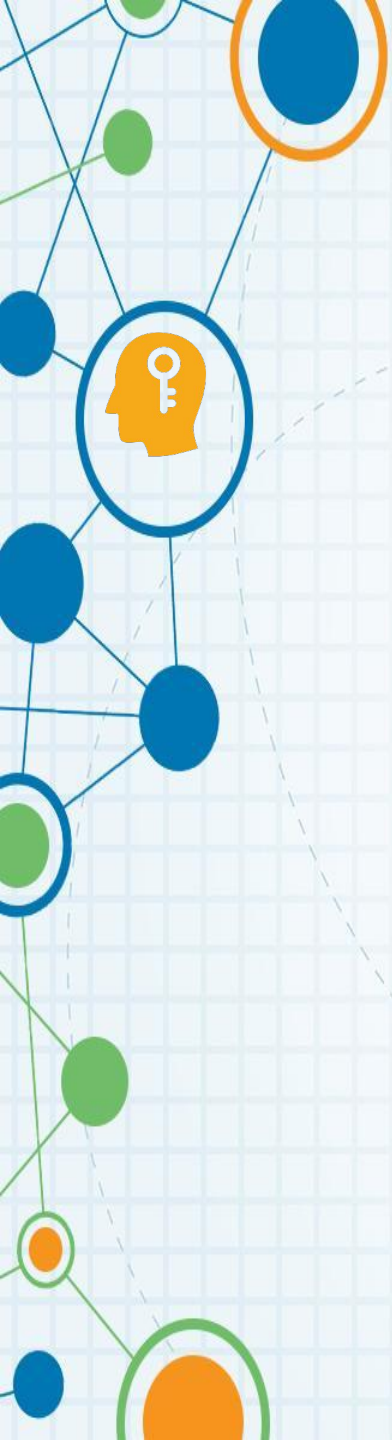


Uptane: Securely Updating Automobiles

Sam Weber | NYU

samweber@nyu.edu

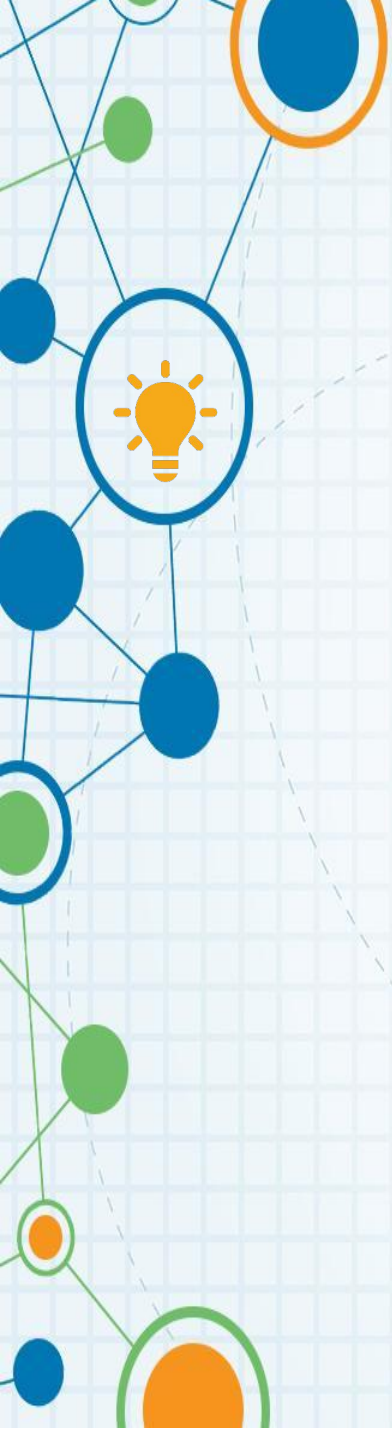
14 June 2017

Credits

- Funded by DHS S&T CSD
- Work done by
  - New York University
  - University of Michigan Transportation Research Institute (UMTRI)
  - Southwest Research Institute (SWRI)

Homeland Security

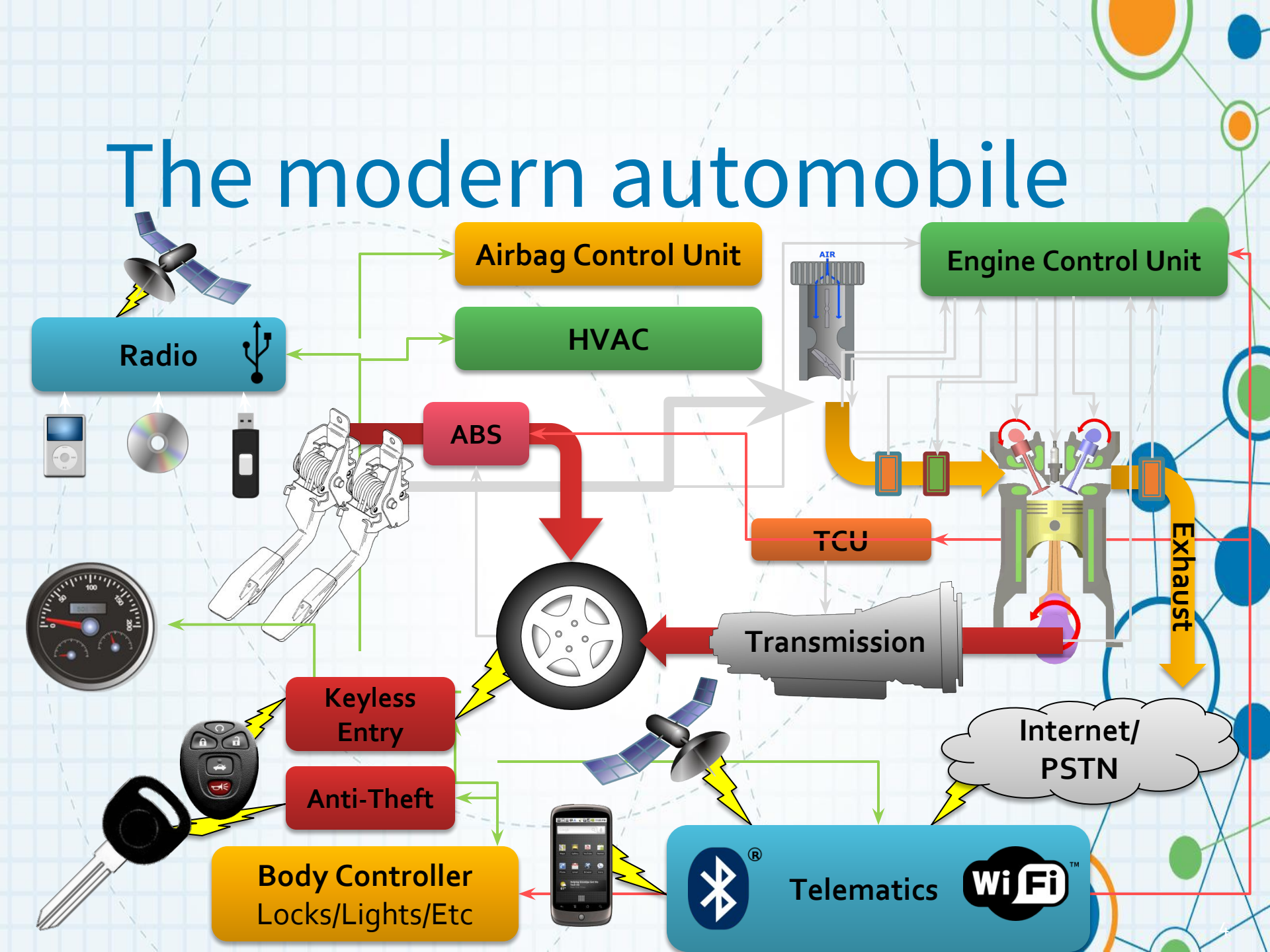
Science and Technology



Overview

- Motivation
- High-level Uptane Design
- Development Process
- Uptane Details
- Summary

The modern automobile



Cars are computerized

- A car is basically a bunch of computers on wheels
  - Complex computerized control
    - Millions of lines of code
    - 50-100 distinct computers (**ECU**s)
  - Shared internal networking (e.g., CAN, FlexRay)
  - Increasing external communications features
    - Telematics, Bluetooth, TPMS, RDS, XM radio, GPS, keyless start/entry, USB ports, WiFi, etc
- Updates both with a mechanic and over-the-air
- Tomorrow's car -> much more of everything
  - V2V/ACAS, V2I, traffic control, autonomous driving

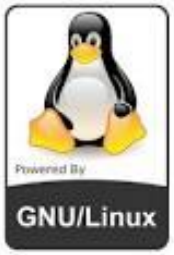
Security Issues

- In 2010 Savage & Kohno's team demonstrated remote attacks on commercial automobiles
  - Could cause brakes to fail while car was on highway
  - Could eavesdrop on conversations in car without occupants being aware

Takeaways

- Cars are very complex cyber-physical systems
- Serious security issues
  - Traditional approaches: recall, or patch flaws ASAP

What do these companies have in common?



Windows

What do these companies have in common?

Victims of a software update hack!

Windows

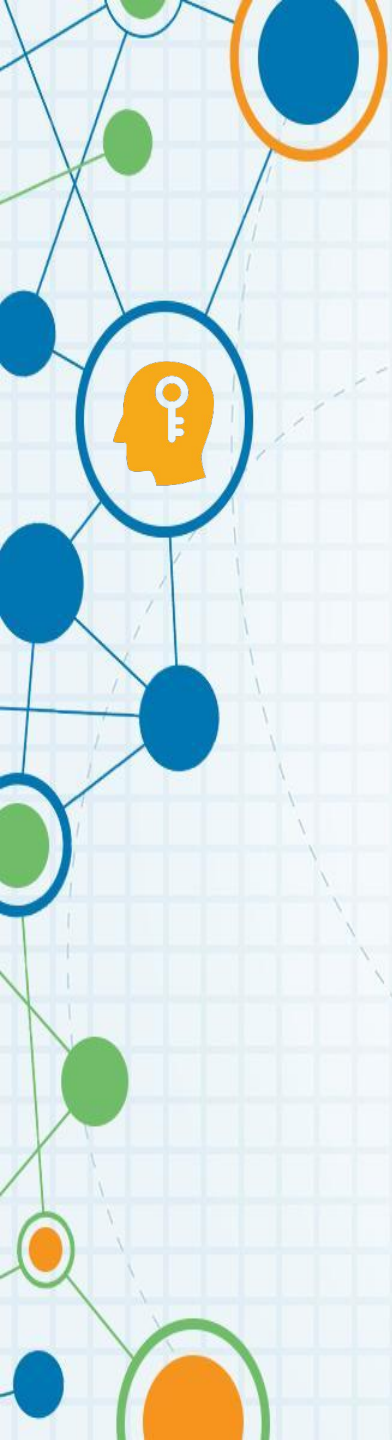
sourceforge

Repository compromise impact

- SourceForge mirror distributed **malware**.
- Attackers impersonate Microsoft Windows Update to spread Flame **malware**.
- RubyGems **compromised** with RCE.
- Opera users **automatically installed malware** signed by **compromised** key.
- Node Packaged Modules **compromised**.
- **Attacks** on software updaters have massive impact
  - E.g. South Korea faced **765 million dollars** in damages.

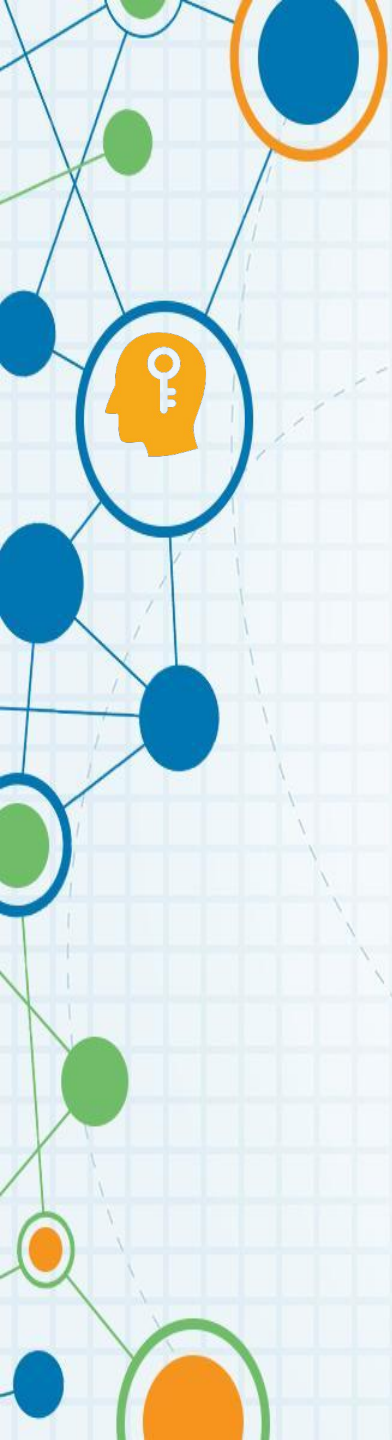
Why are so many SUs vulnerable?

- Lack of separation of trust
  - All eggs in one basket
- Assume repository / key will never be compromised



Software Updater

- The purpose is to fix a flaw
- An insecure updater means you have **two** flaws



High-level Design

Principles

- Separation of Duties
  - Each entity does only one thing
  - Limits effect of attack
- Ability to recover from adverse event
  - Ex: revoke compromised keys
- Address known threats

Threat Examples

- Forging update messages to car
- Attacked update service:
  - distributes malware-infected ECU images
  - distributes valid but incompatible ECU images
  - downgrades ECUs to versions with known vulnerabilities

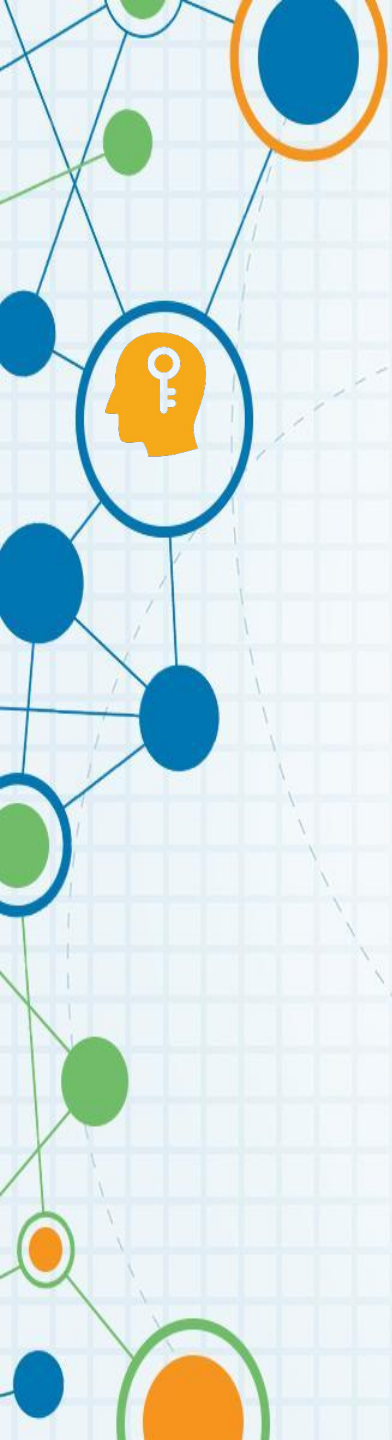
Main Entities

Server-Side:

- Director Repository
  - Informs car of what updates needed
- Image repository
  - Holds valid ECU images

Car:

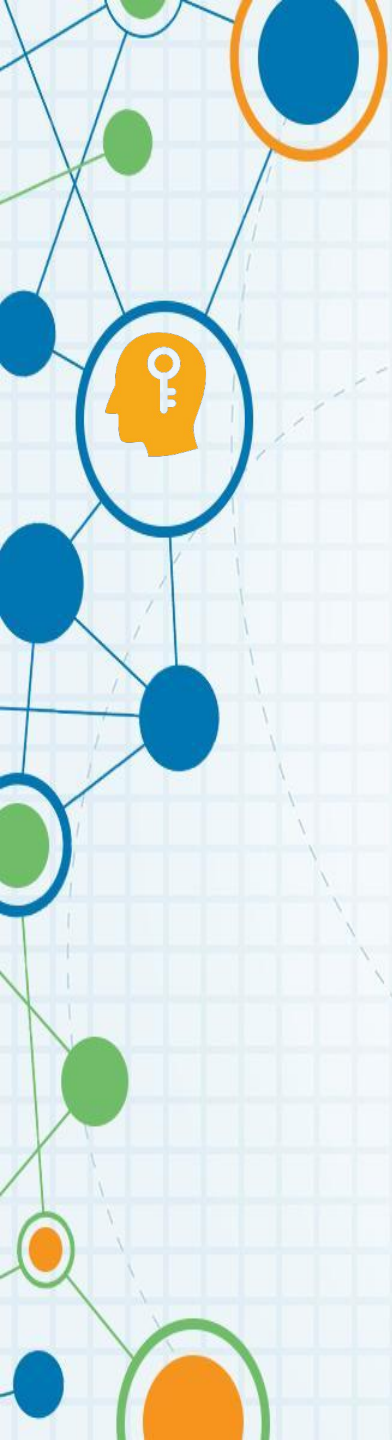
- Primary ECU(s)
  - Responsible for communication outside car, internal distribution of updates
- Secondary ECUs
  - update themselves, possibly relay on internal network



Development Process

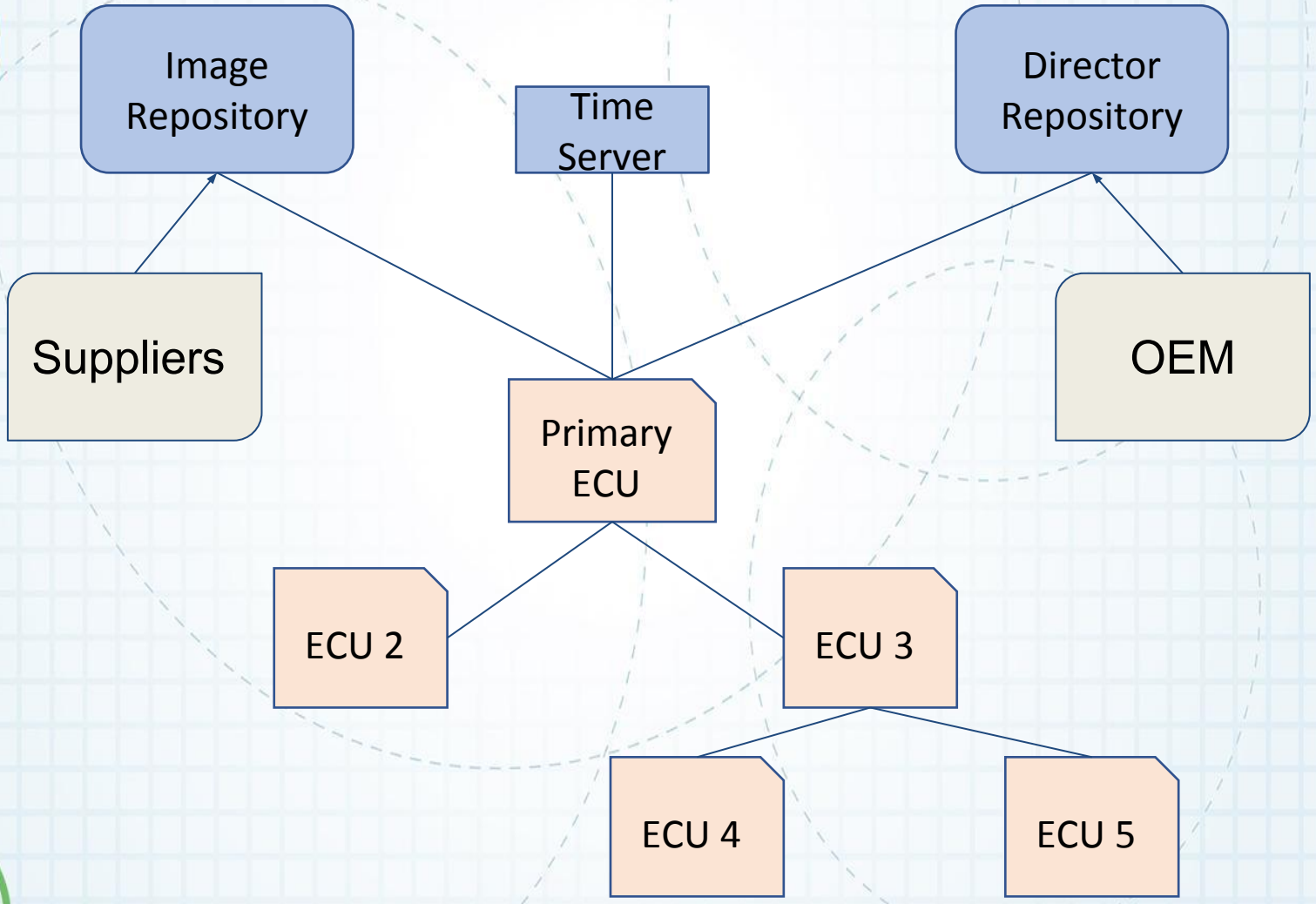
Process

- Engaging with automotive community
  - 40+ participants (OEMs, tier 1s, gov)
- Openly available:
  - Detailed draft specification, deployment recommendations
  - Reference implementation
- Current activities:
  - Open security review
  - Discussions with T1s on Uptane integration
  - Potential use by Commercial Truck industry

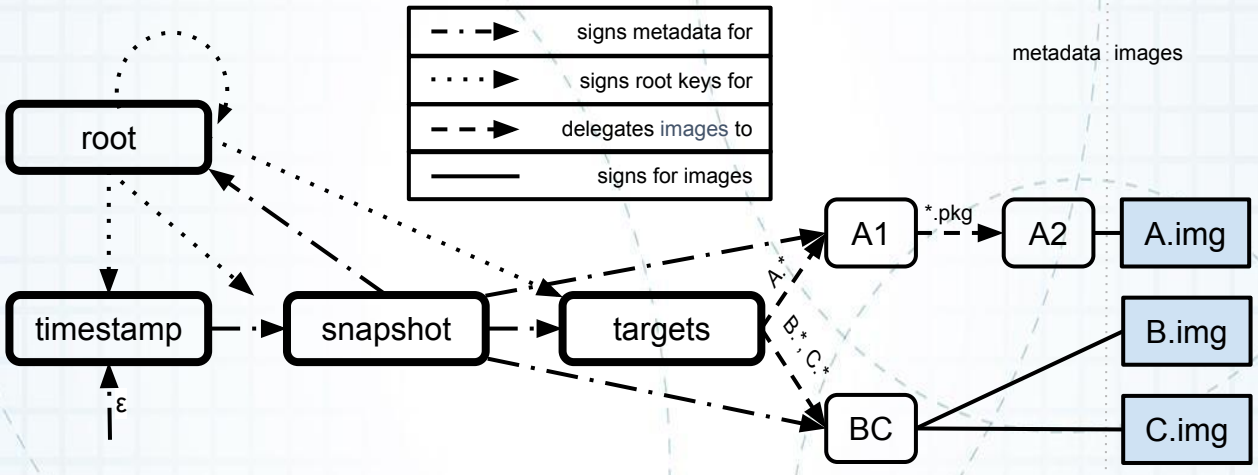


Detailed Design

Entities and Communication



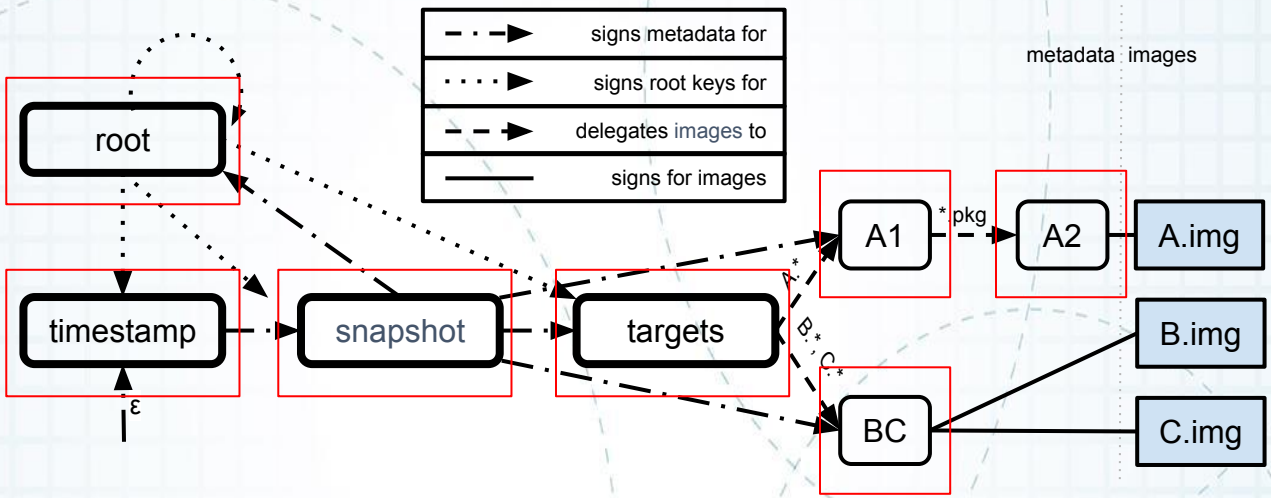
Uptane: Design Principles



Design principles:

1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. Minimize key / role risk.

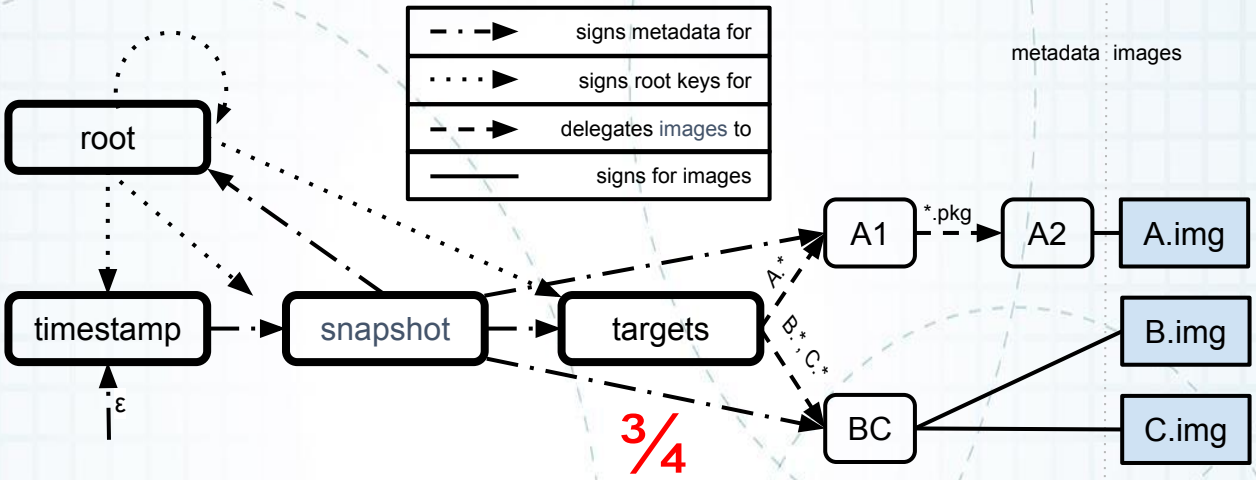
Separation of Duties



Design principles:

1. **Separation of duties.**
   - Sign different types of metadata using different keys.
   - Metadata about images (self-contained archives of code+data for ECUs), or other metadata files.

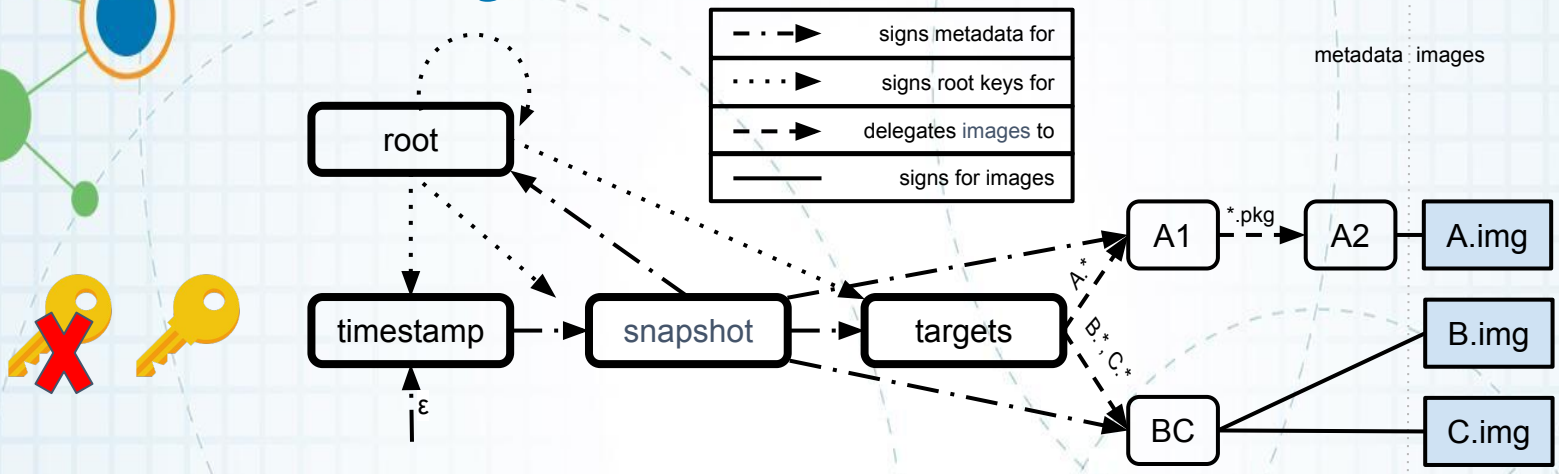
Threshold signatures



Design principles:

1. Separation of duties.
2. **Threshold signatures.**

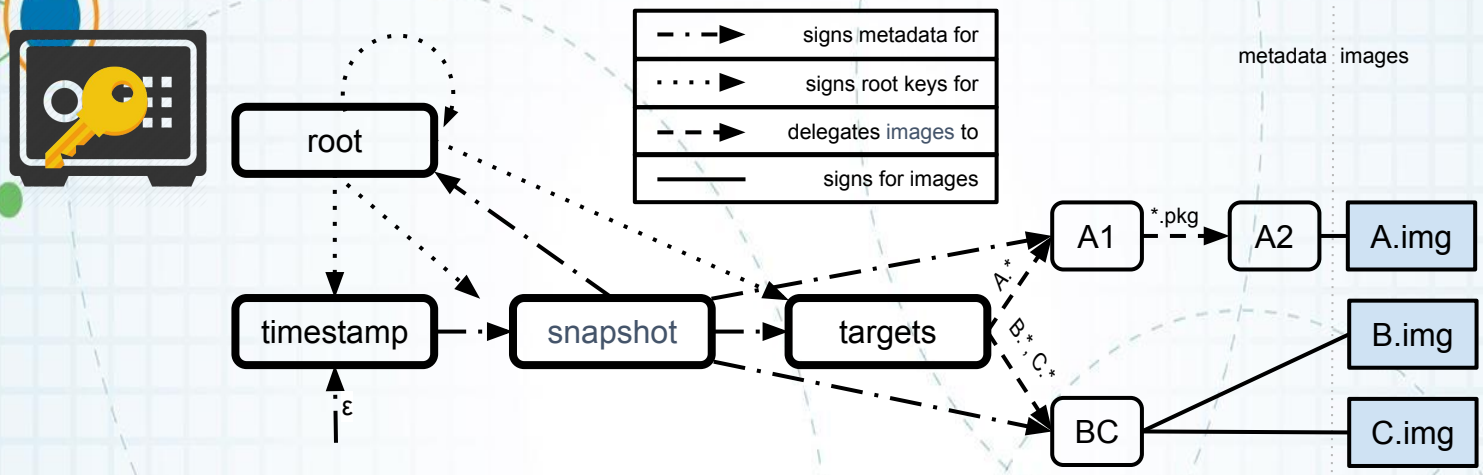
Explicit and Implicit Key Revocation



Design principles:

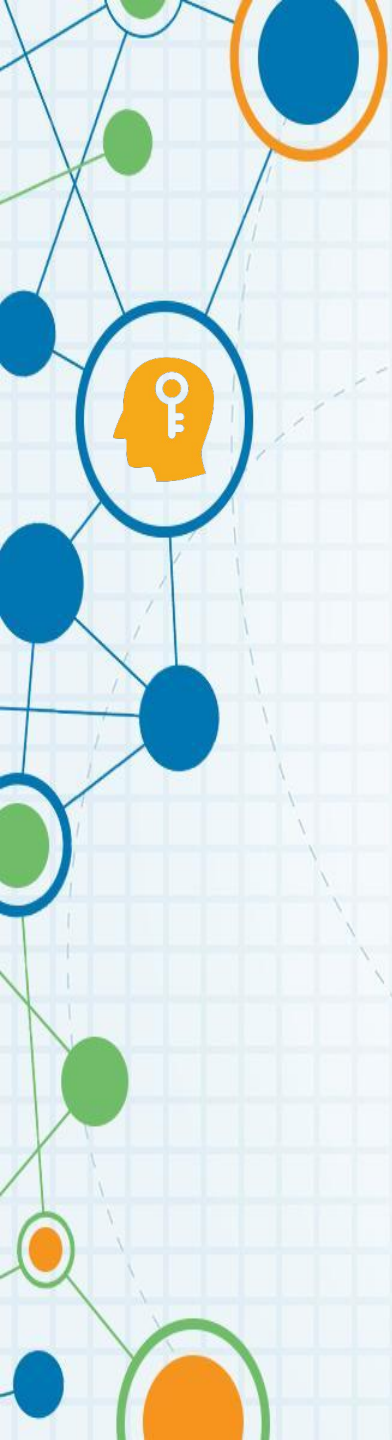
1. Separation of duties.
2. Threshold signatures.
3. **Explicit and implicit revocation of keys.**

Minimizing Risk



Design principles:

1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. **Minimize key / role risk.**



Summary

- Automotive updates involve many non-obvious issues
- Uptane working with automobile and security communities
  - open design process
- Draft specification and other documents available

See https://uptane.github.io

Contacts

Website:

https://uptane.github.io

(Public mailing list, and documents available there)