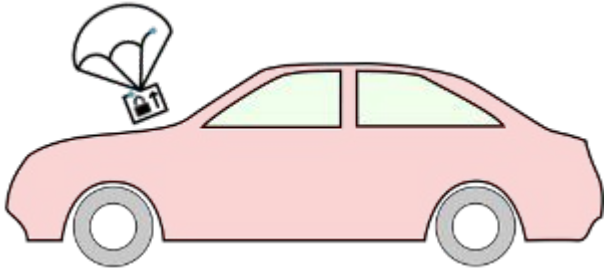
Uptane

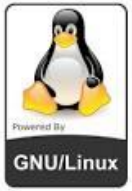
Securing Over-the-Air Updates
Against Nation State Actors



Justin Cappos
New York University
uptane.github.io

NYU TANDON SCHOOL OF ENGINEERING

What do these companies have in common?



What do these companies have in common?




Users attacked via software updater!

Windows

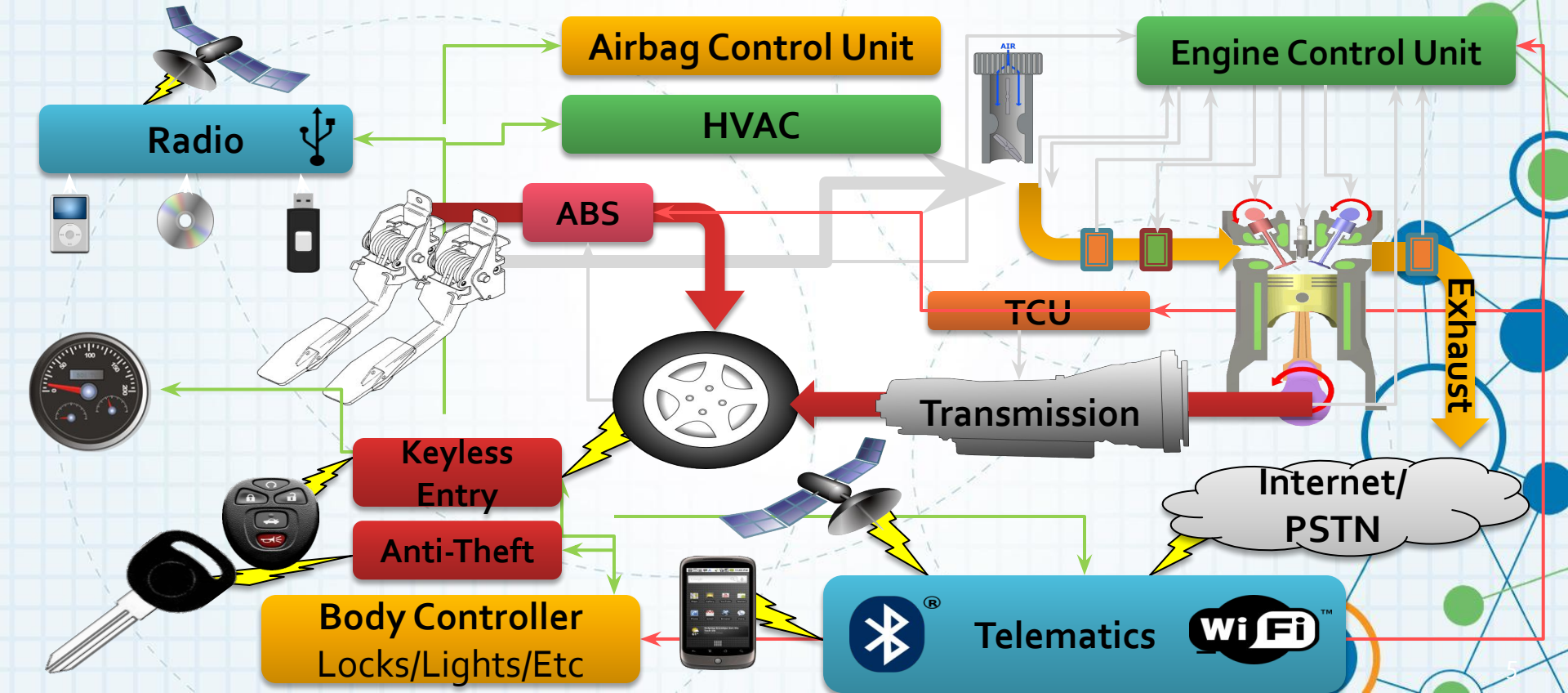
sourceforge

GitHub

Software repository compromise impact

- SourceForge mirror distributed **malware**.
- Attackers impersonate Microsoft Windows Update to spread Flame **malware**.
- Attacks on software updaters have massive impact
  - E.g. South Korea faced 765 million dollars in damages.
- NotPetya spread via software updates!

The modern automobile



Cars Are Dangerous

- Researchers have made some scary attacks against vehicles
  - remotely controlling a car's brakes and steering while it's driving
  - spontaneously applying the parking brake at speed
  - turning off the transmission
  - locking driver in the car

Cars are multi-ton, fast-moving weapons

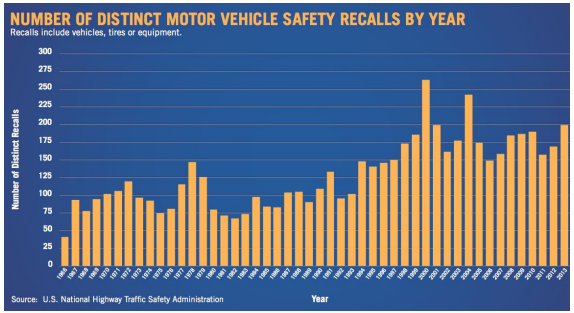
People will die

Updates Are Inevitable

- Millions of lines of code means bugs
- Regulations change -> firmware must change
- Maps change
- Add new features
- Close security holes
- Cars move across borders…

Updates Must Be Practical

- Updating software/firmware has often meant recalls.
- Recalls are extremely expensive
  - GM spent $4.1 billion on recalls in 2014
  - GM's net income for 2014 was < $4 billion
  - People do not like recalls.
- Updates must be over the air.



Updates Are Dangerous

- Update -> Control

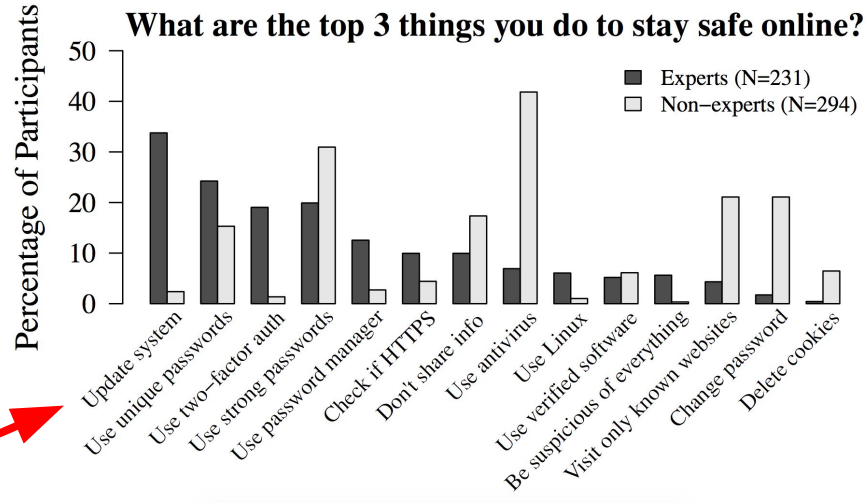
Secure Updates

- Nation-state actors pull off complex attacks
  - Must not have a single point of failure

What to do?

Must update to fix security issues

Insecure update mechanism is a new security problem

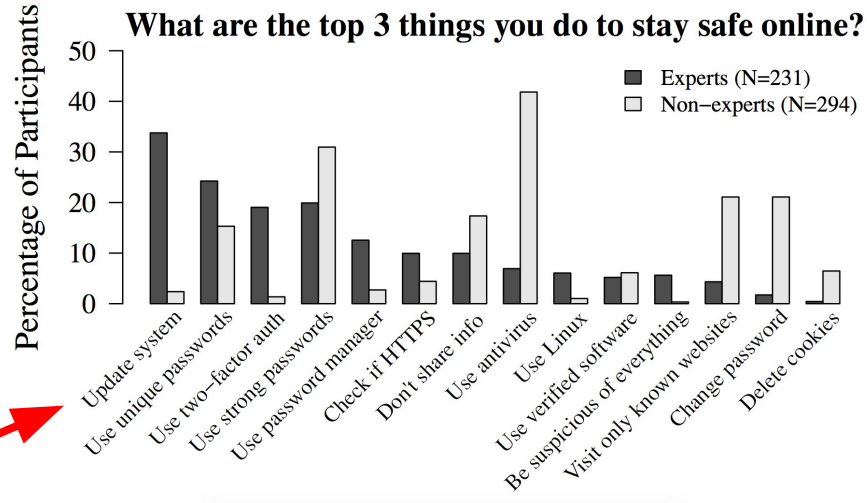


"...No one Can Hack My Mind": Comparing Expert and Non-Expert Security Practices
Ion, et al. SOUPS 2015

Security Defense Types

Must update to fix security issues

Insecure update mechanism is a new security problem



"...No one Can Hack My Mind": Comparing Expert and Non-Expert Security Practices
Ion, et al. SOUPS 2015

Attacks

- **Arbitrary software installation**. An attacker can provide arbitrary files in response to download requests and install anything he wants on the client system, yet none will be detected as illegitimate.
- **Rollback attacks**. An attacker presents files to a software update system that are older than those the client

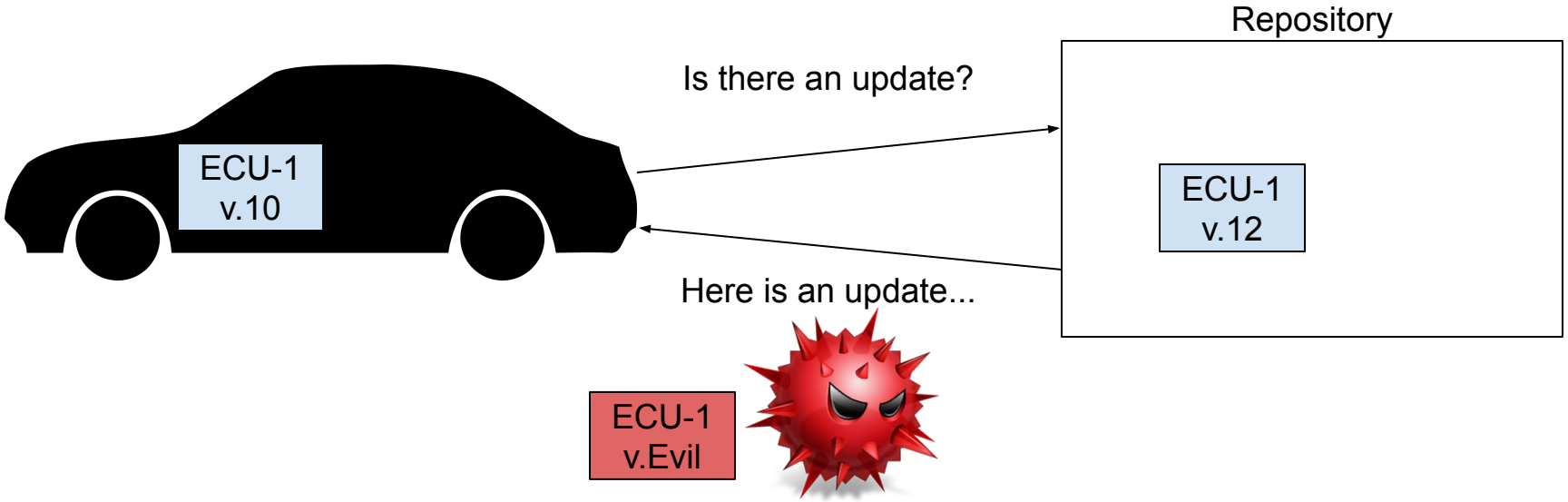
Attack Name	Description	Requirement	Result	Rule
Slow Retrieval	An attacker slows r… and will not error o…	Repository	DoS	(1)
Endless Data	A malicious reposi… any file request.	Repository	DoS / Crash	(1)
Replay Old Metadata	An attacker provid… packages from bein…	Repository	Outdated Package	(1)
Extraneous Dependency	An attacker change… or packages of the …	Metadata Key	Any Signed Package	(2)
Depends on Everything	An attacker change…	Metadata Key	DoS / Crash	(2)
Unsatisfiable Dependencies	An attacker causes … data indicates unsat…	Metadata Key	DoS / Out-dated Package	(2)
Provides Everything	An attacker change… dency the user requ…	Metadata Key	Any Signed Package	(2)
Use Revoked Keys	An attacker uses a r…	Revoked Key	Arbitrary Package	(3)
Escalation of Privilege	An attacker compro… then gets users to a…	Package Key	Arbitrary Package	(3)

Icon	Security attack
	Eavesdrop attack
	Drop-request attack
	Freeze attack
	Partial bundle installation attack
	Rollback attack
	Endless data attack
	Mixed-bundles attack
	Mix-and-match attack
	Arbitrary software attack

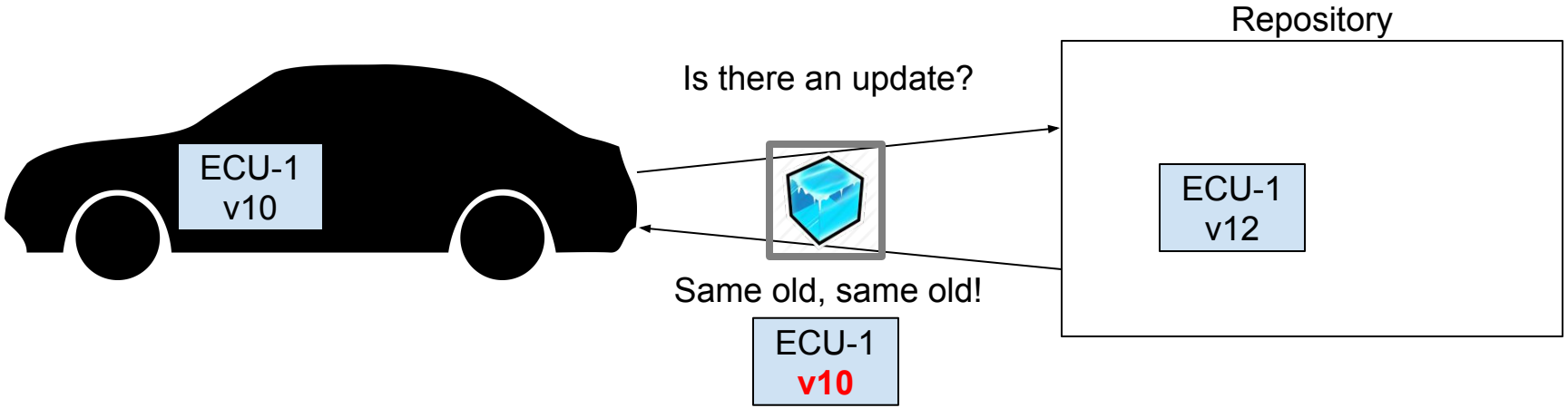
prevent clie…

- **Vulnerability to key compromises**. An attacker who can compromise the one key in a single key system, or less than a given threshold of keys, can compromise clients. These attacks can occur whether the client relies on a single online key (if only being protected by SSL) or a single offline key (if protected by most software update systems that use keysigning).

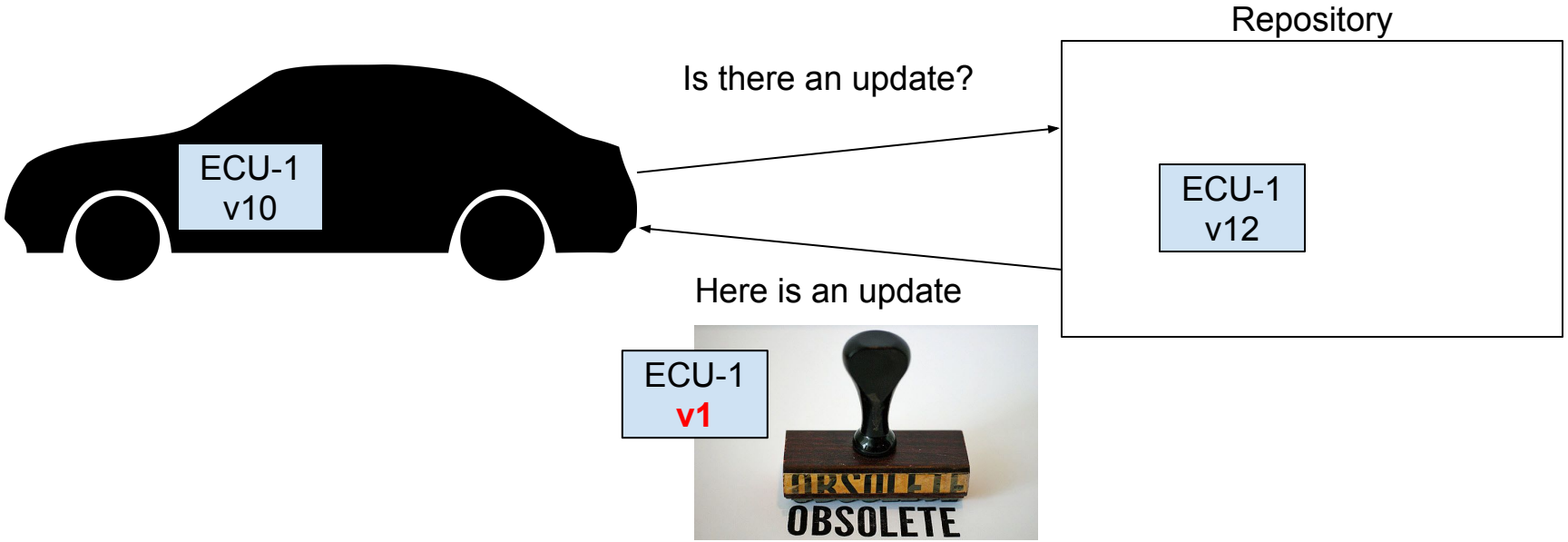
Arbitrary software attack



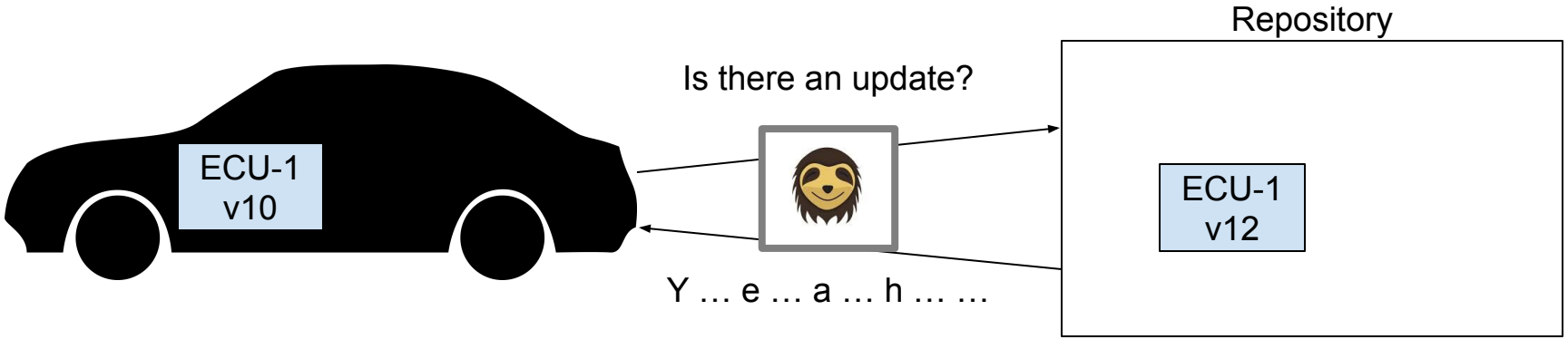
Freeze attack



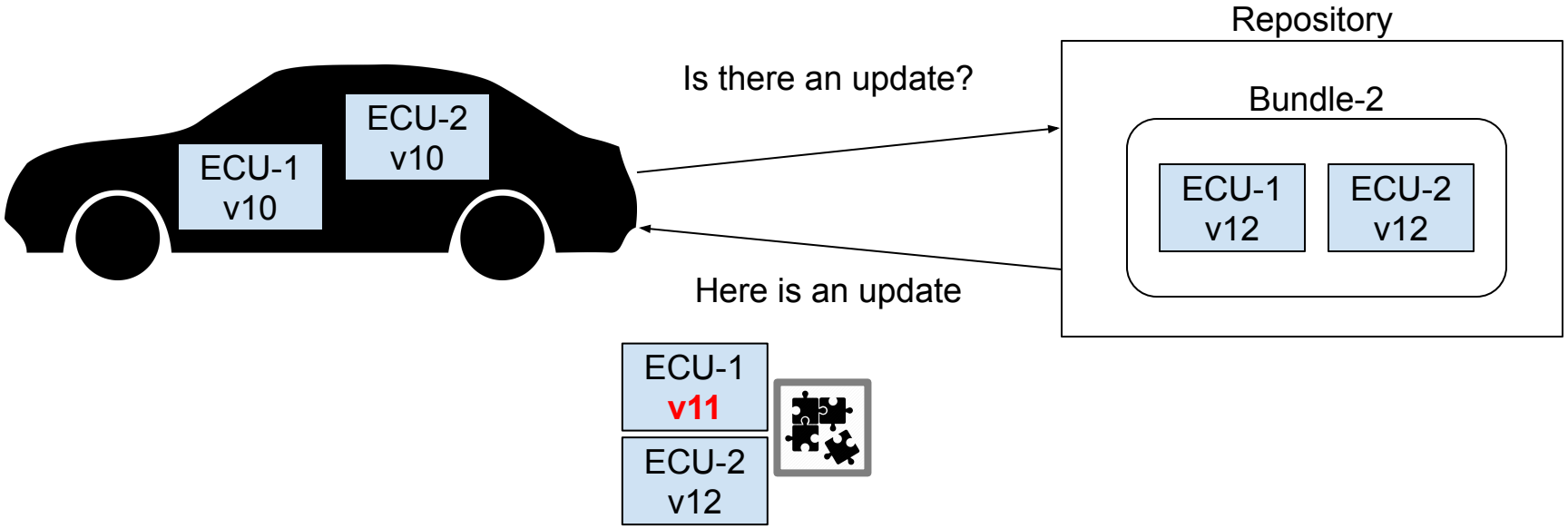
Rollback attack



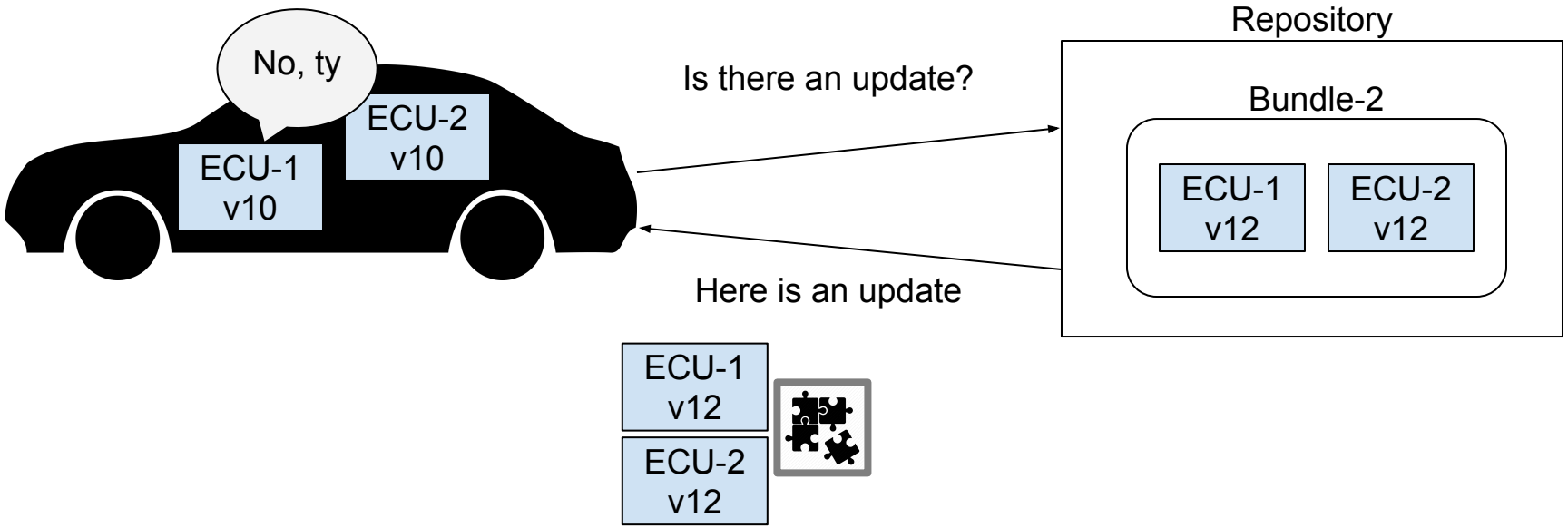
Slow retrieval attack



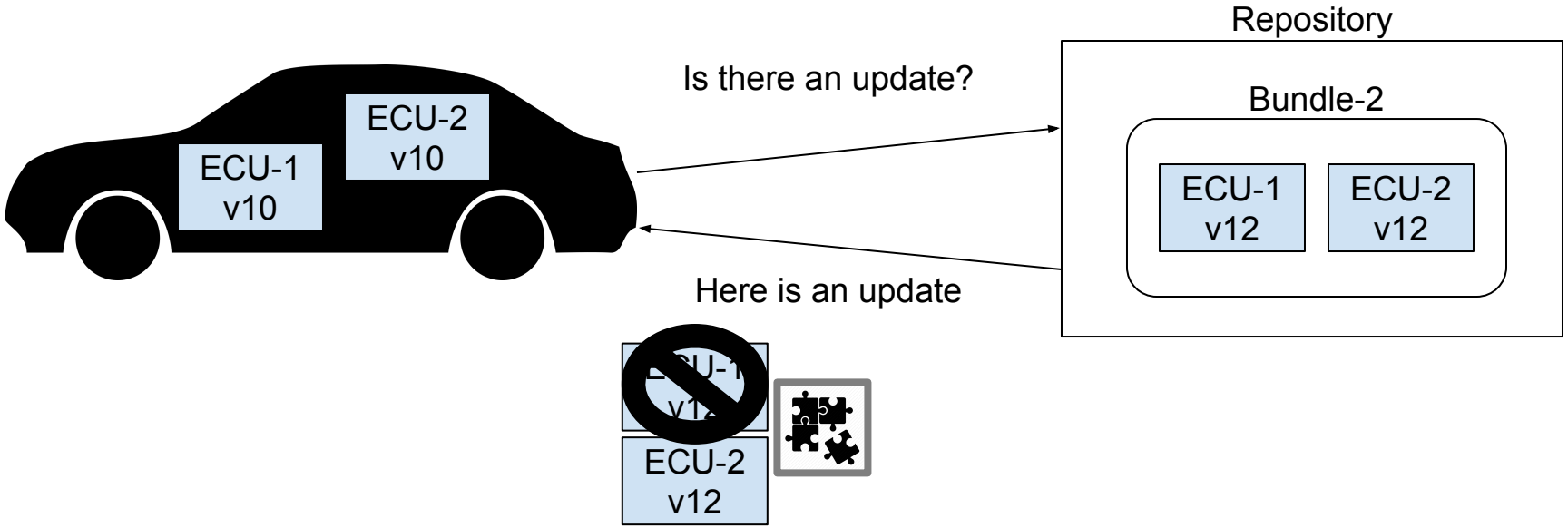
Mix and Match attacks



Partial Bundle attack



Partial Freeze attack

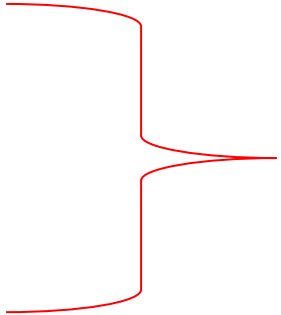


How to address security concerns

- Prevent
  - Make it harder for a compromise to occur
- Detect
  - Detect incidents of compromise quickly
- Transfer Risk
  - Have insurance or claim regulations were followed
- Mitigate
  - Make a successful compromise less impactful

How to address security concerns

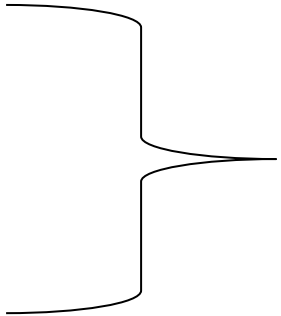
- Prevent
  - Make it harder for a compromise to occur
- Detect
  - Detect incidents of compromise quickly
- Transfer Risk
  - Have insurance or claim regulations were followed
- Mitigate
  - Make a successful compromise less impactful



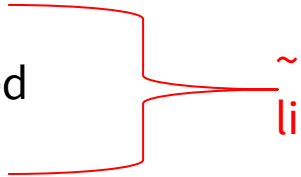
Most automotive technologies

How to address security concerns

- Prevent
  - Make it harder for a compromise to occur
- Detect
  - Detect incidents of compromise quickly
- Transfer Risk
  - Have insurance or claim regulations were followed
- Mitigate
  - Make a successful compromise less impactful



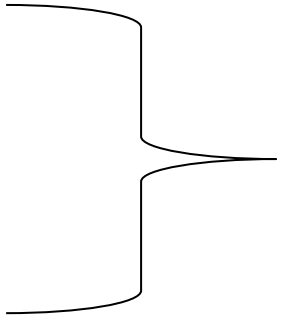
Most automotive technologies



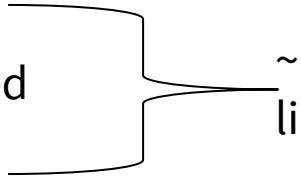
~100Ms USD lawsuit, likely unachievable

How to address security concerns

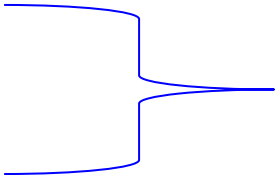
- Prevent
  - Make it harder for a compromise to occur
- Detect
  - Detect incidents of compromise quickly
- Transfer Risk
  - Have insurance or claim regulations were followed
- Mitigate
  - Make a successful compromise less impactful



Most automotive technologies



~100Ms USD lawsuit, likely unachievable



Major Uptane value add

How to address security concerns

- Prevent
  - Make it ha
- Detect
  - Detect inc
- Transfer Risk
  - Have insurance or claim regulations were followed
- Mitigate
  - Make a successful compromise less impactful

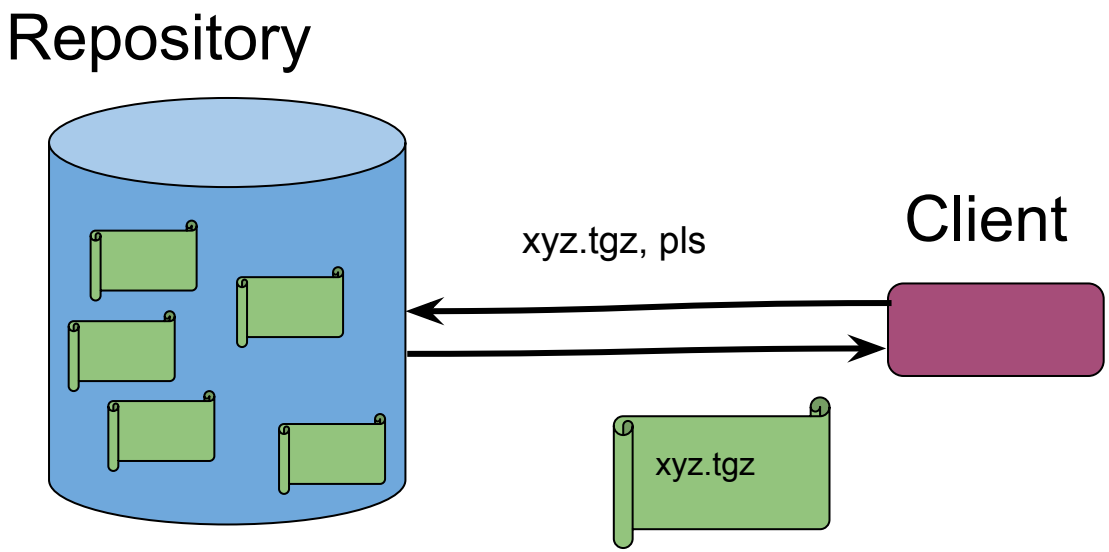
OMA-DM, ITU-T X.1373, etc. enable full control with a single compromise

st automotive
hnologies

~100Ms USD lawsuit, likely unachievable

Major Uptane value add

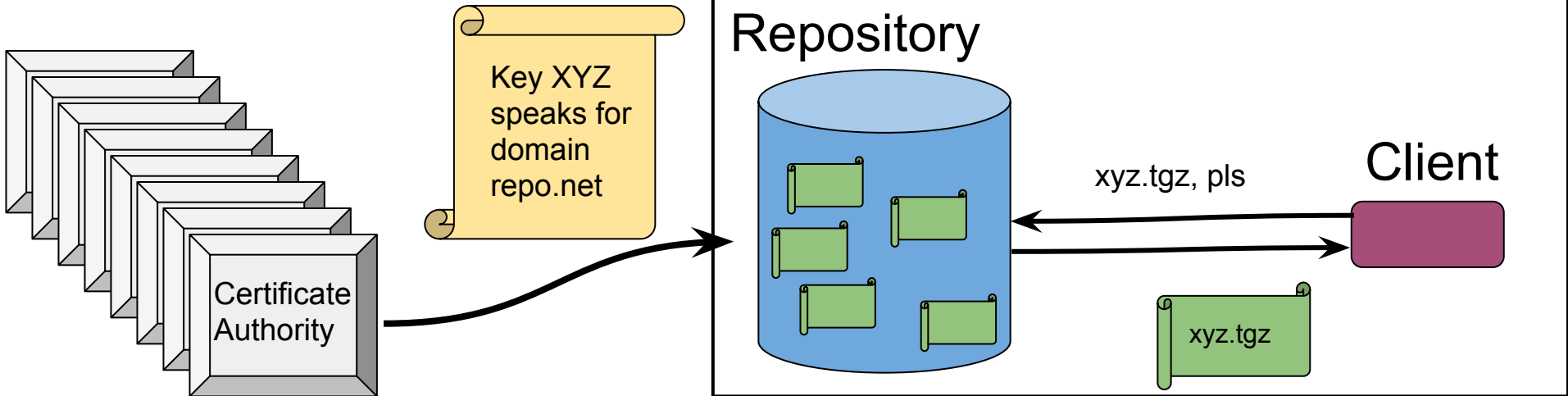
Update Basics



Inadequate Update Security 1: TLS/SSL

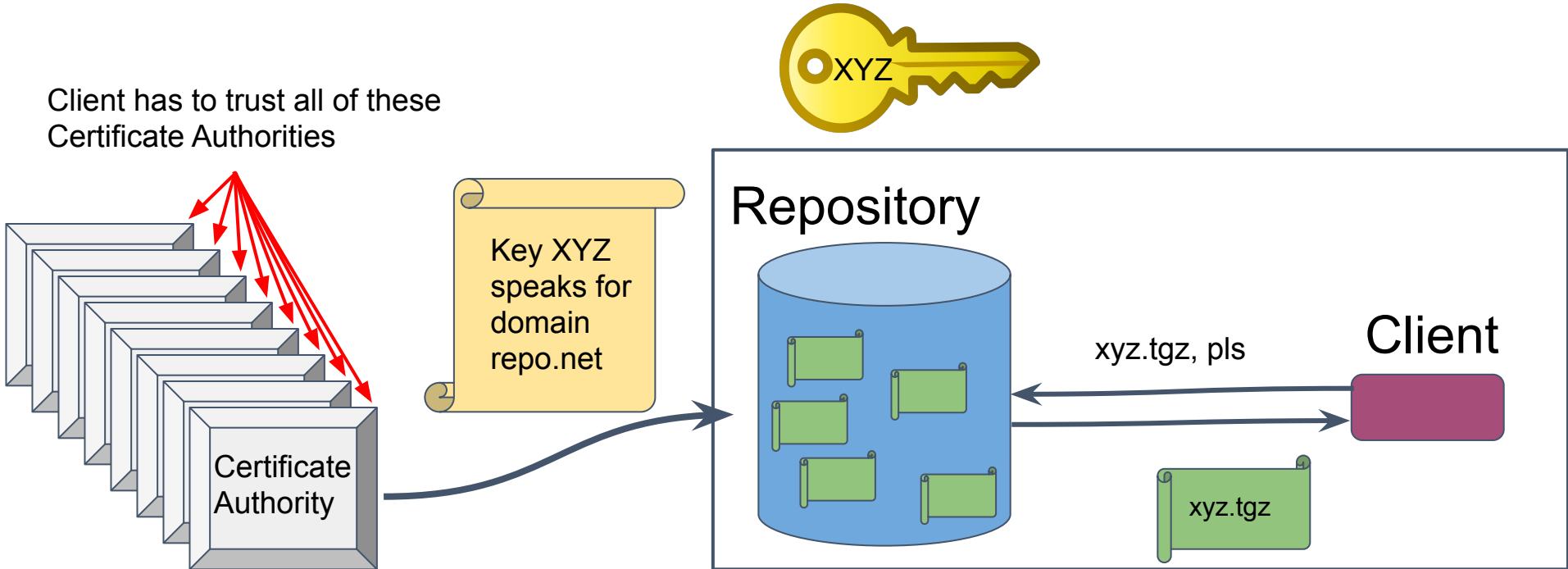
Traditional solution 1:

Authenticate the repository (TLS, SSL, etc)



Inadequate Update Security 2: TLS/SSL

Transport Layer Security: Problem 1

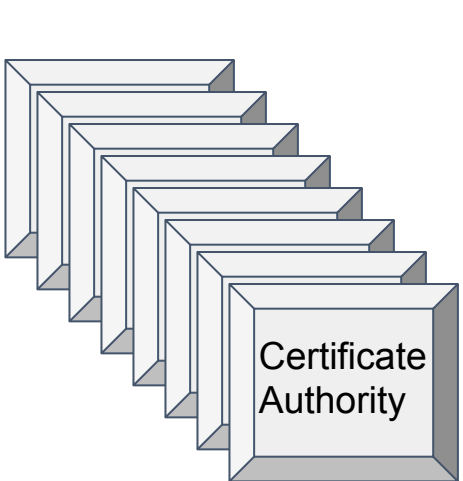


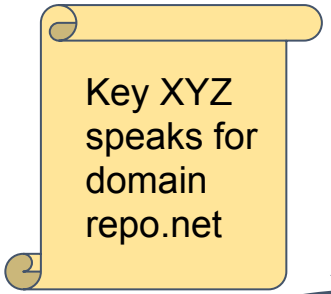
Inadequate Update Security 3: TLS/SSL

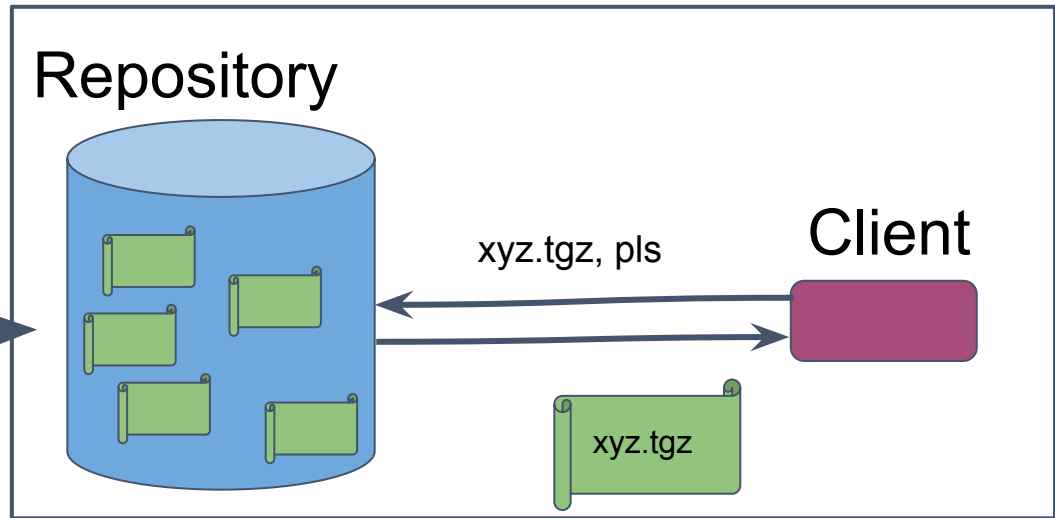
Transport Layer Security: Problem 2

Client has to trust this key.

… which HAS to exist ON the repository, to sign communications continuously.



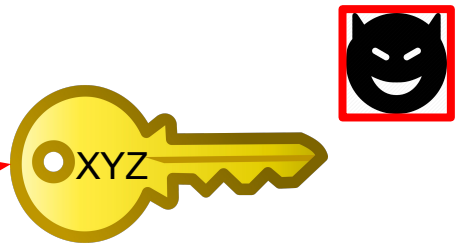




Inadequate Update Security 4: Just Sign!

Traditional Solution 2:

Sign your update package with a specific key.
Updater ships with corresponding public key.

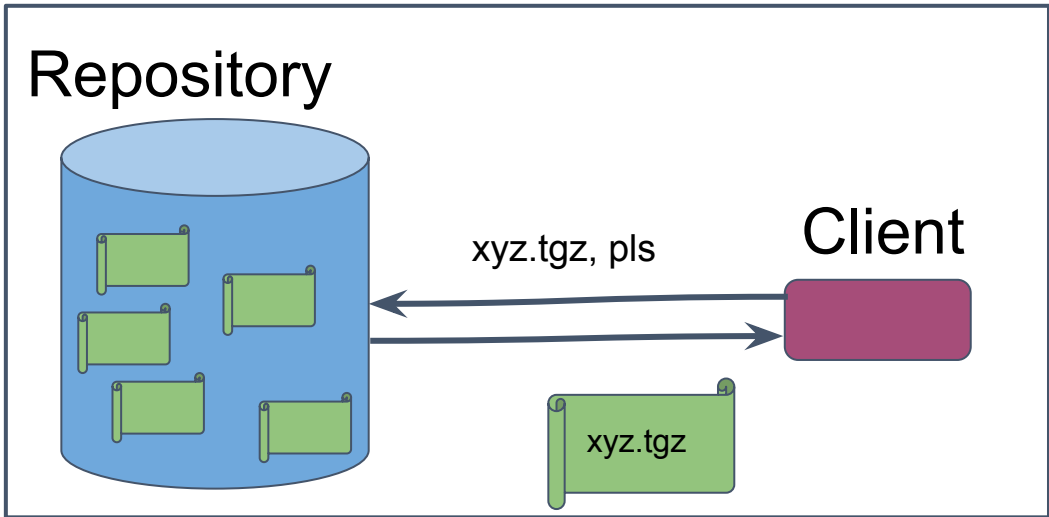


Client has to trust this key

… used for every update to the repository.

… key ends up on repo or build farm.

If an attacker gains the use of this key, they can install arbitrary code on any client.

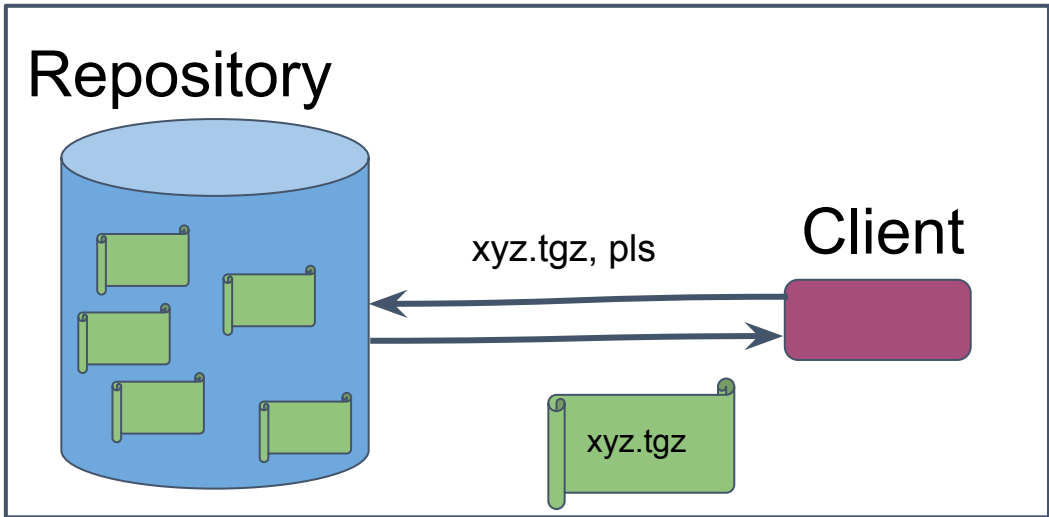


Update Security

We need:

- To survive server compromise with the minimum possible damage.
  - Avoid arbitrary package attacks
- Minimize damage of a single key being exposed
- Be able to revoke keys, maintaining trust
- Guarantee freshness to avoid freeze attacks
- Prevent mix and match attacks
- Prevent rollback attacks
- Prevent slow retrieval attacks
- ...

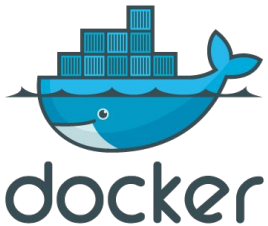
Must not have single point of failure!



The Update Framework (TUF)

Linux Foundation CNCF project

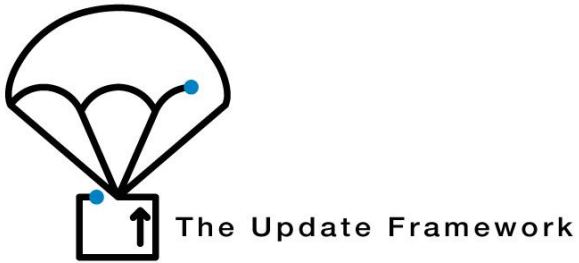
Widely used in industry:



The Update Framework (TUF): Goals

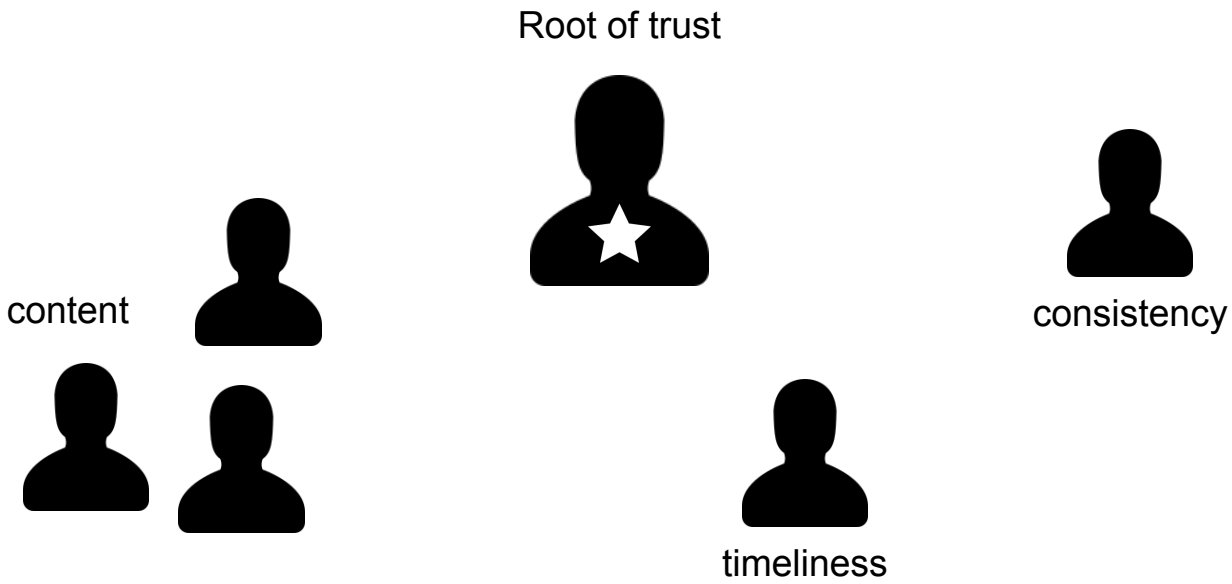
TUF goal “Compromise Resilience”

- TUF secures software update files
- TUF emerges from a serious threat model:
  - We do NOT assume that your servers are perfectly secure
  - Servers will be compromised
  - Keys will be stolen or used by attackers
  - TUF tries to minimize the impact of every compromise



The Update Framework (TUF)

Responsibility Separation

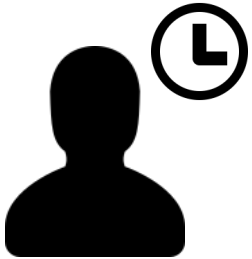


The Update Framework (TUF)

TUF Roles Overview

Root

(root of trust)

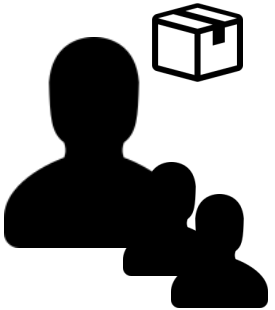


Timestamps

(timeliness)

Snapshot

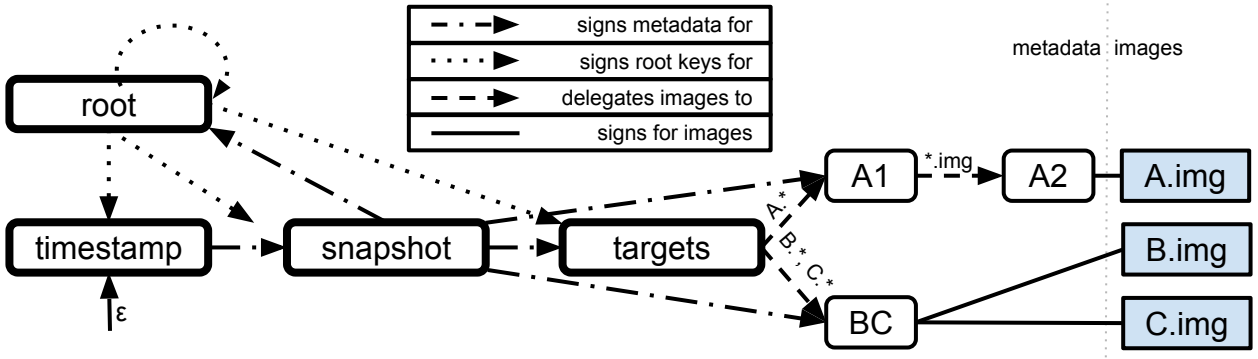
(consistency)



Targets

(integrity)

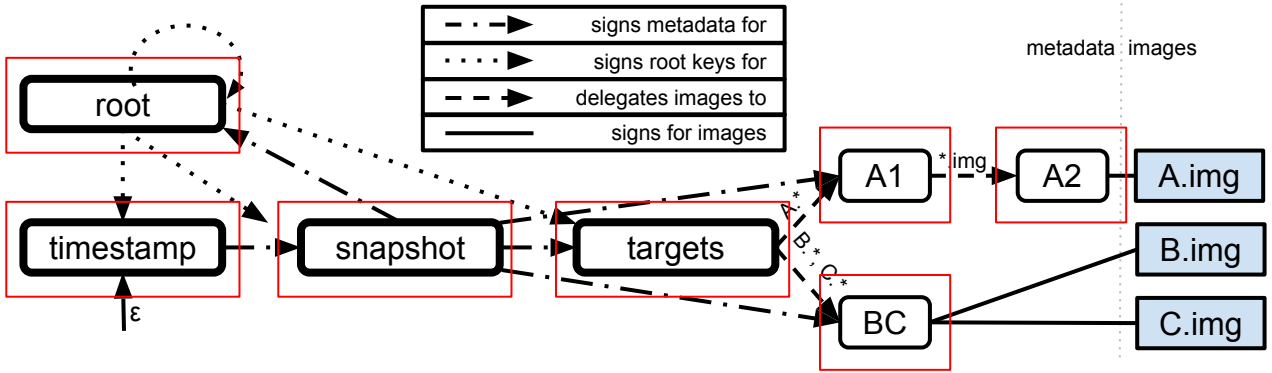
Design principles for a repository



Design principles:

1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. Minimized risk through use of offline keys.

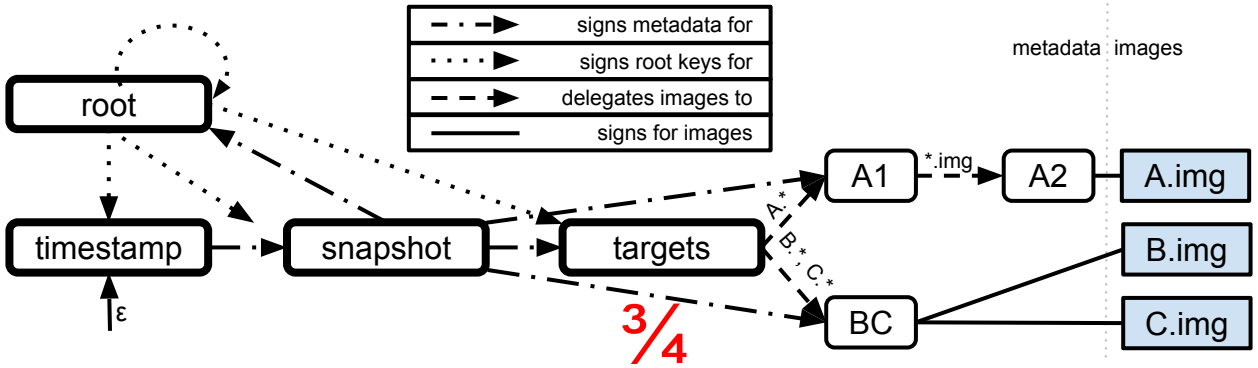
Separation of duties



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.

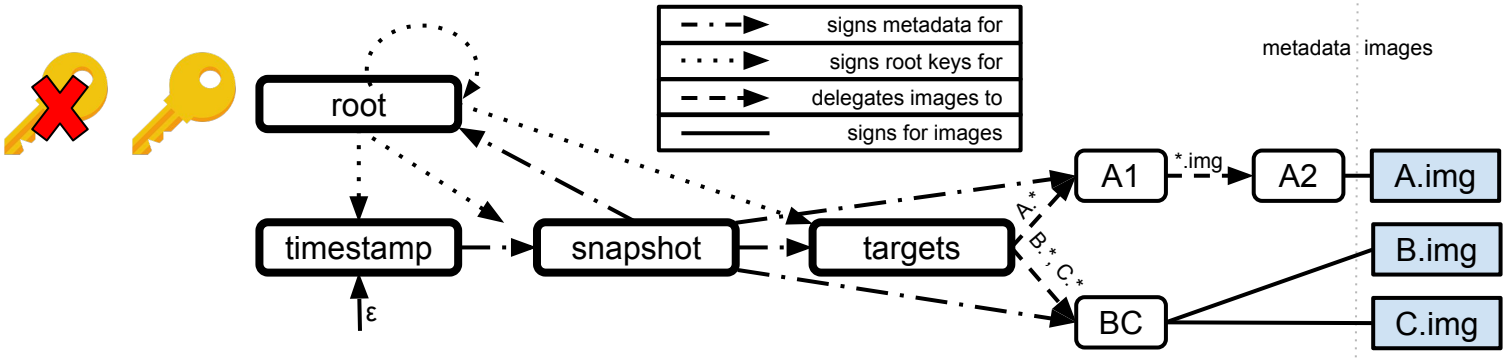
Threshold signatures



Design principles:

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

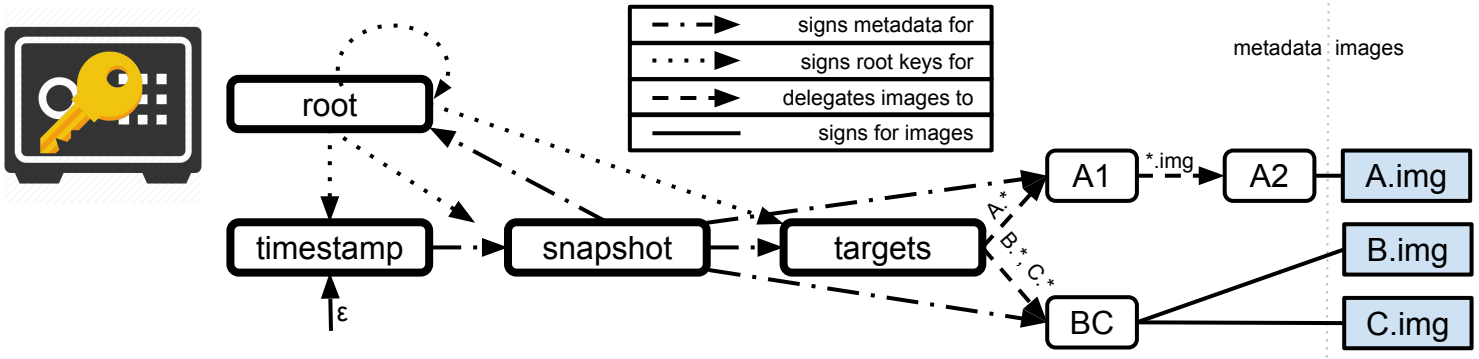
Explicit & implicit revocation of keys



Design principles:

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

Minimizing risk with offline keys

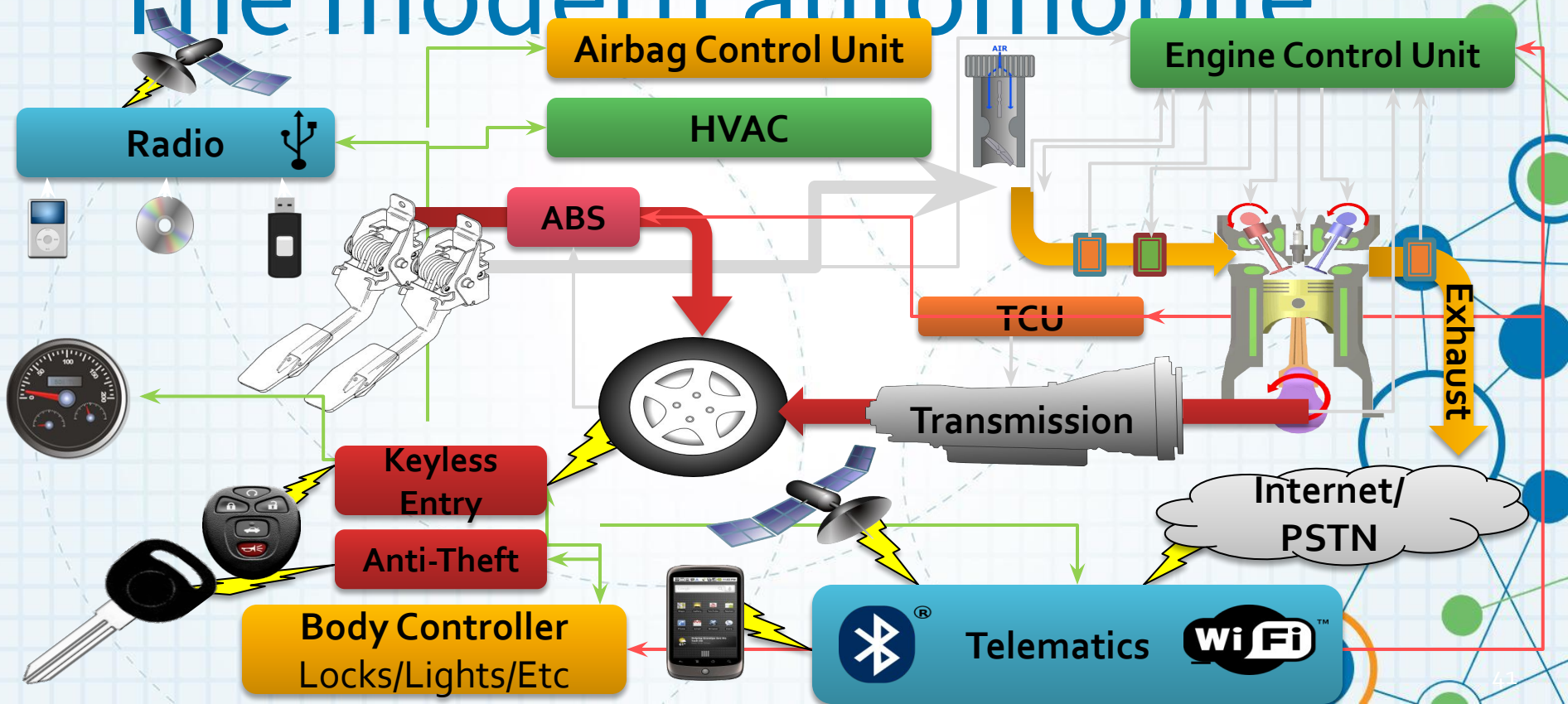


Design principles:

1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. **Minimized risk through use of offline keys.**

Automobiles present particular difficulties.

The modern automobile



Uptane builds on The Update Framework (TUF)

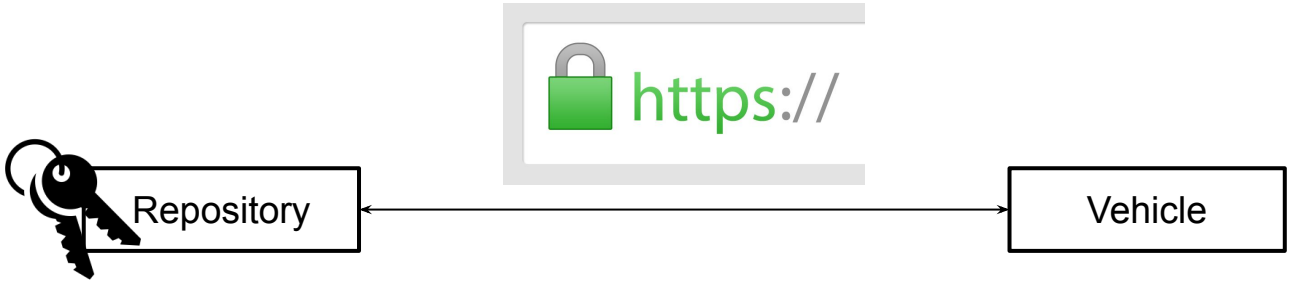
- Timeserver
- Multiple Repositories: Director and Image Repository
- Manifests
- Primary and Secondary clients
- Full and Partial verification

Background

- Repository contains images + metadata
- Image
  - A unit of update
  - An archive of code + data for an ECU
  - One image per ECU
- Metadata
  - Information such as cryptographic hashes and file sizes
  - About images, or other metadata files

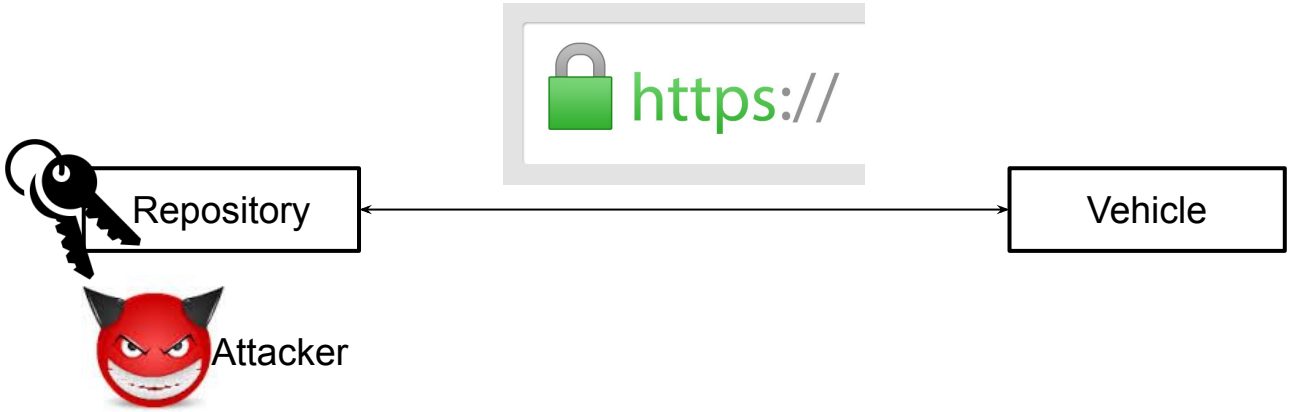
Signing all metadata with an online key

- Use a single online key to sign all metadata (e.g., using SSL / TLS)
- Protects ECUs from man-in-the-middle attacks between repository and vehicle
- Allows on-demand customization of updates for vehicles



The problem with an online key

- Doesn’t say anything about the security of the server: just that you are talking to it
- Single point of failure: easy to compromise
- If repository is compromised, attacker can install malware and control vehicles

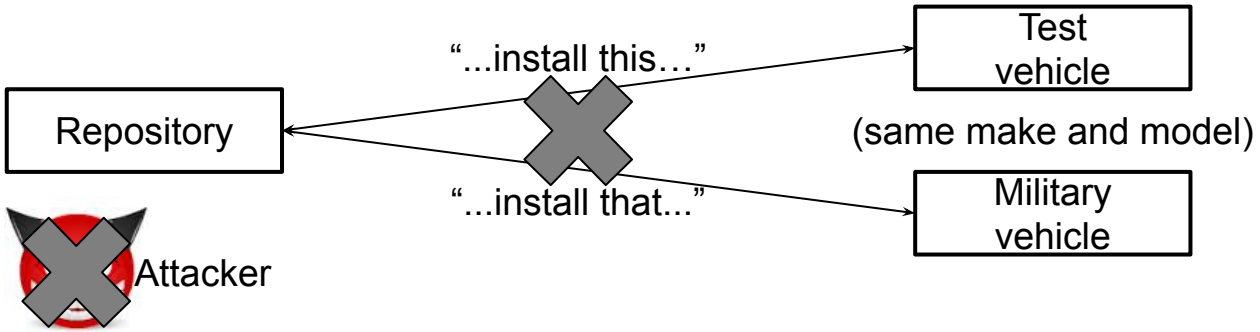


Signing all metadata with an offline key

- Use a single offline key to sign all metadata (e.g., using GPG or RSA)
- Compromise-resilient, because attackers cannot tamper with metadata without being detected

The problem with an offline key

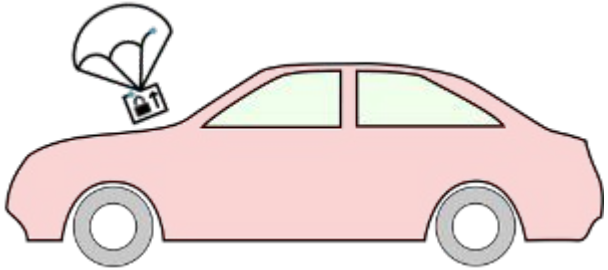
- Difficult to customize updates on-demand for vehicles
  - Difficult to install different updates on vehicles of the same make and model, but with different requirements
  - Cannot instantly blacklist only buggy updates
- In practice, this risks becoming the previous system (online key)



Takeaway: either-or

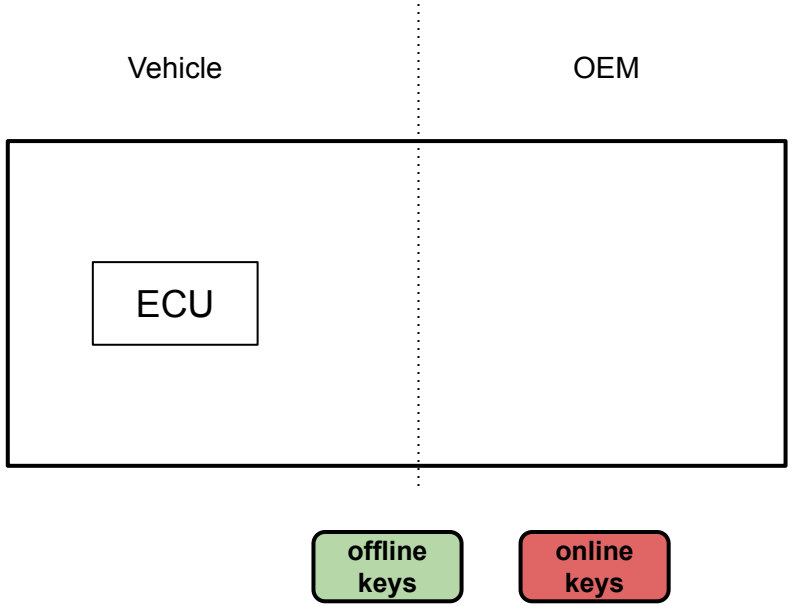
- Previous security systems force repositories to choose either on-demand customization of vehicles, or compromise-resilience.

Avoiding either-or security choices



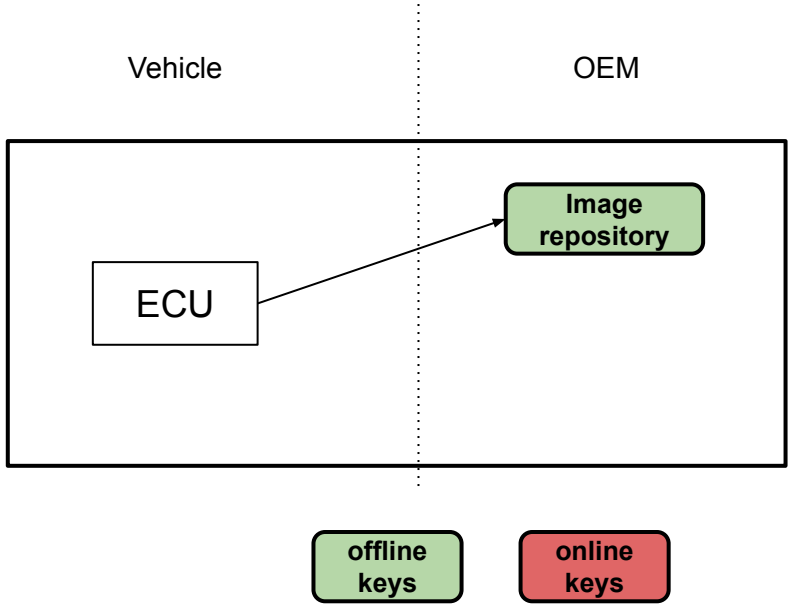
Key idea

- What if there are two repositories?



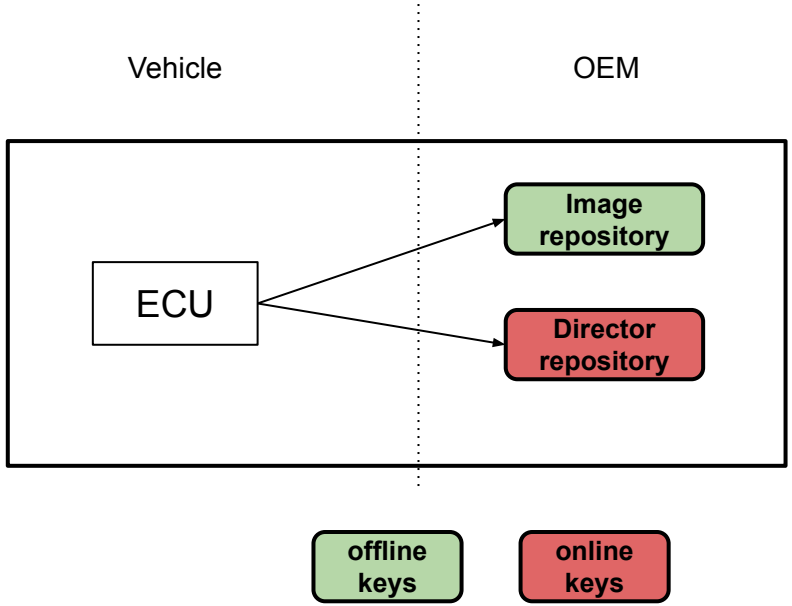
Key idea

- What if there are two repositories?
- Image repository
  - Uses offline keys
  - Provides signed metadata about all available updates for all ECUs on all vehicles



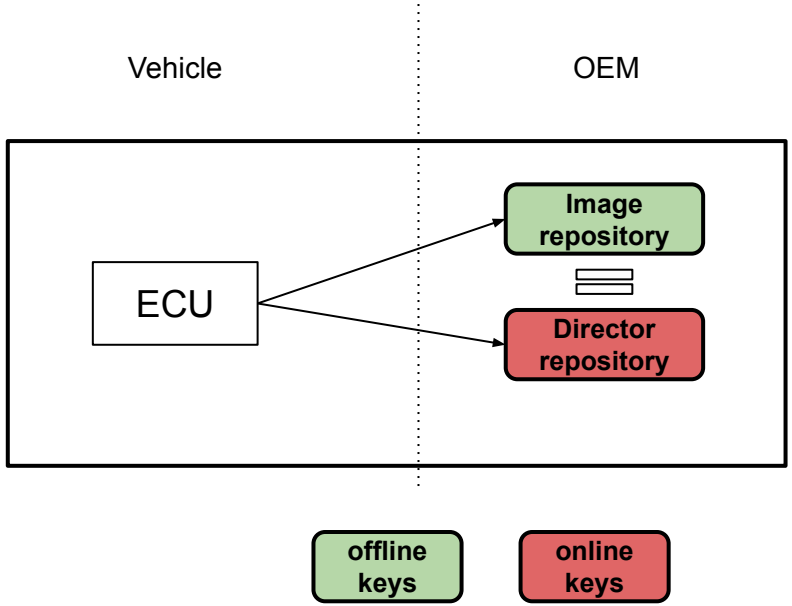
Key idea

- What if there are two repositories?
- Image repository
  - Uses offline keys
  - Provides signed metadata about all available updates for all ECUs on all vehicles
- Director repository
  - Uses online keys
  - Signs metadata about which updates should be installed on which ECUs on a vehicle

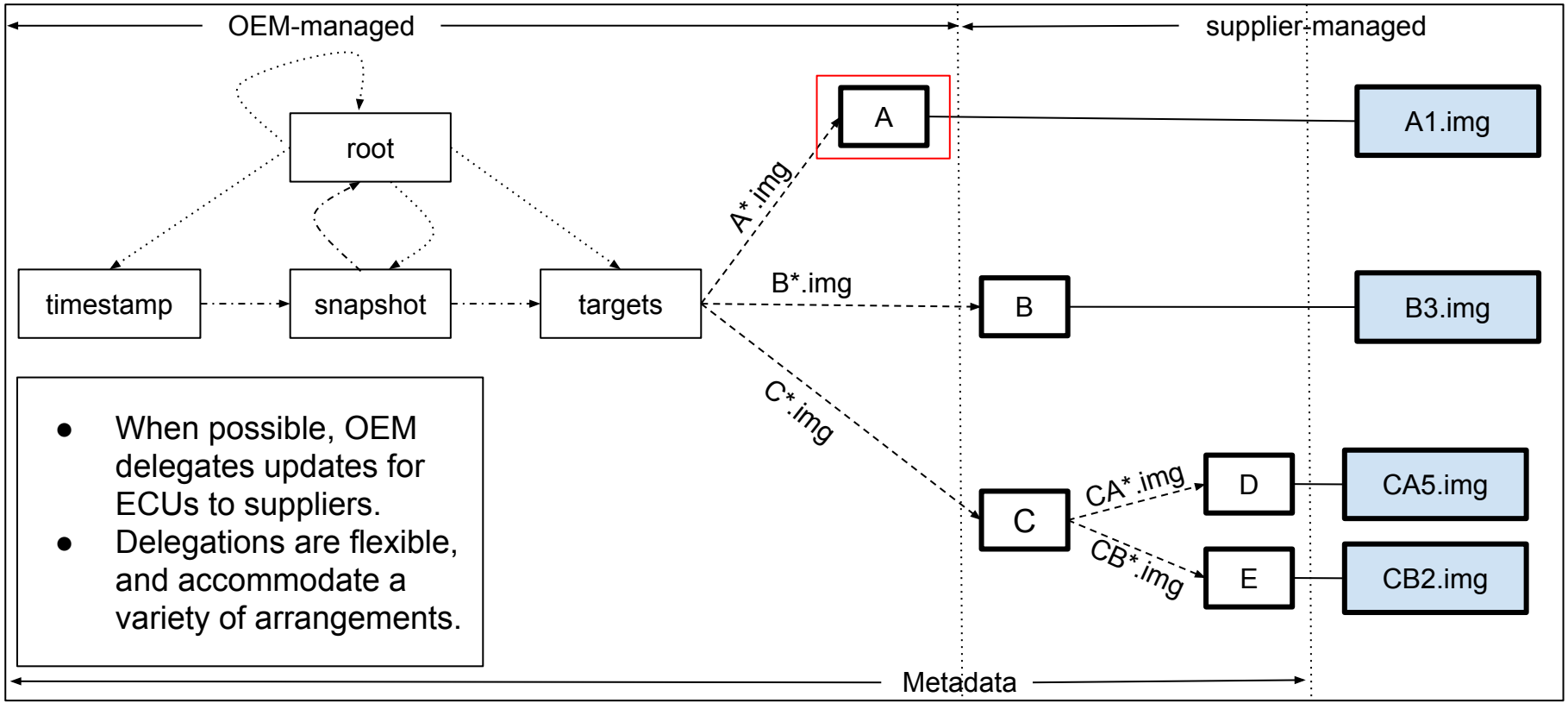


Key idea

- A vehicle would ensure that installation instructions from the director repository matches updates from the image repository.
- Using both repositories provides both on-demand customization of vehicles & compromise-resilience.

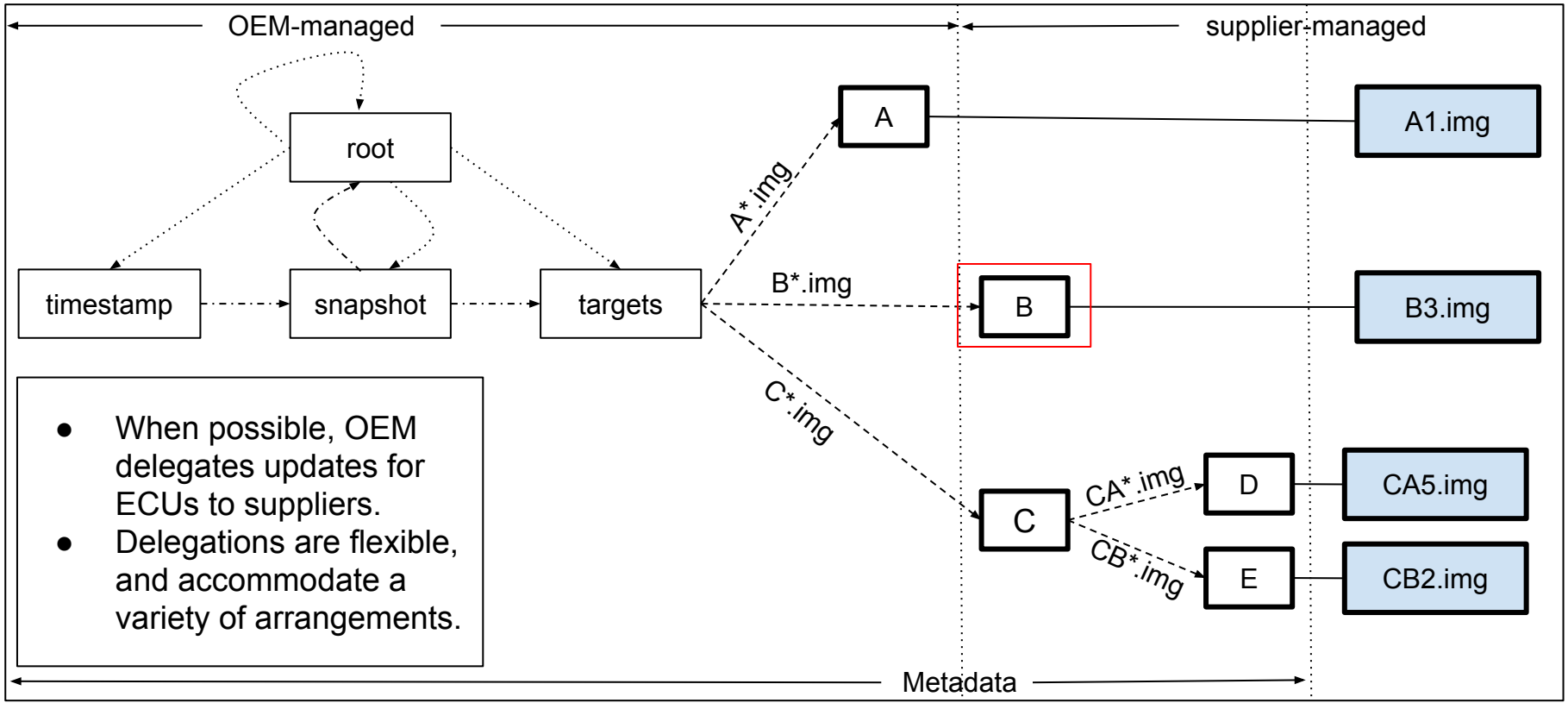


The image repository



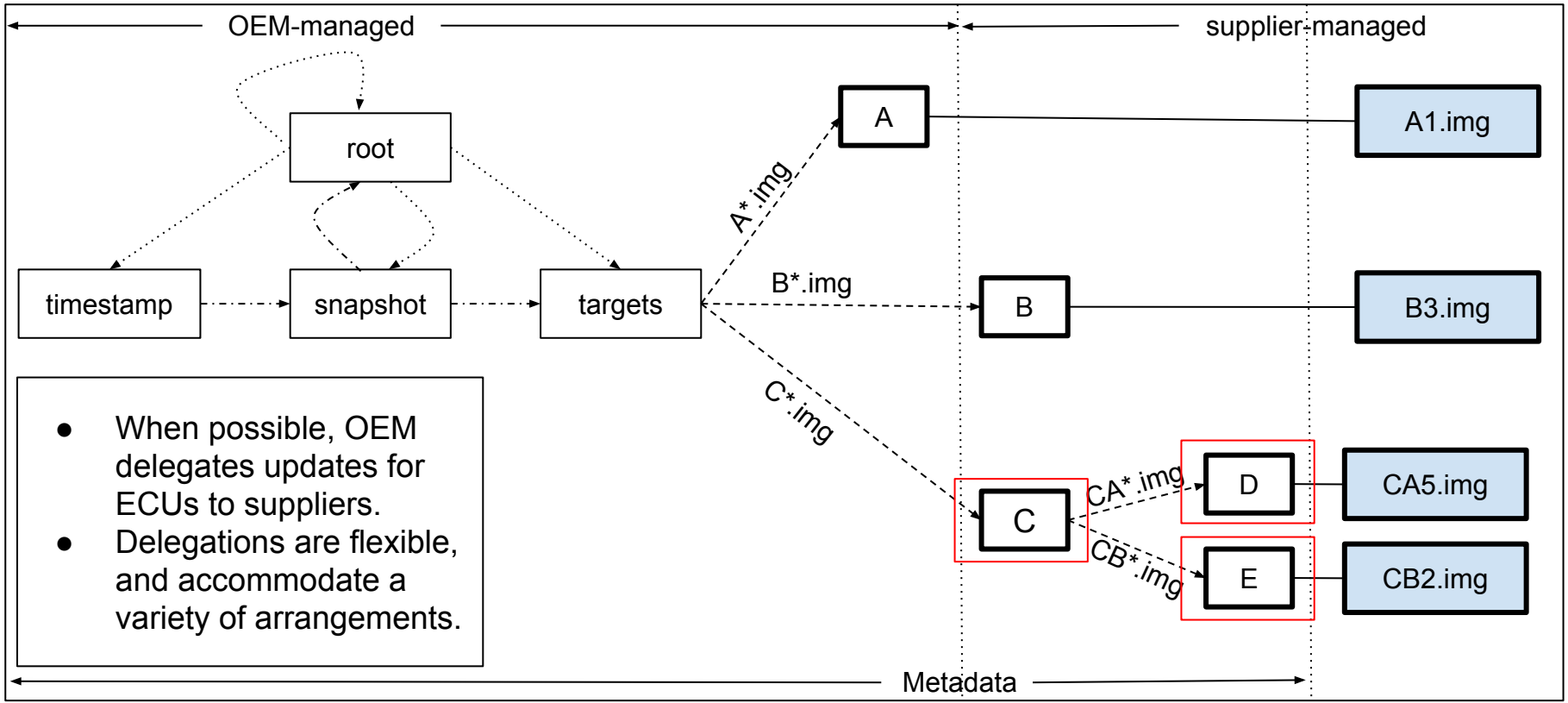
- When possible, OEM delegates updates for ECUs to suppliers.
- Delegations are flexible, and accommodate a variety of arrangements.

The image repository



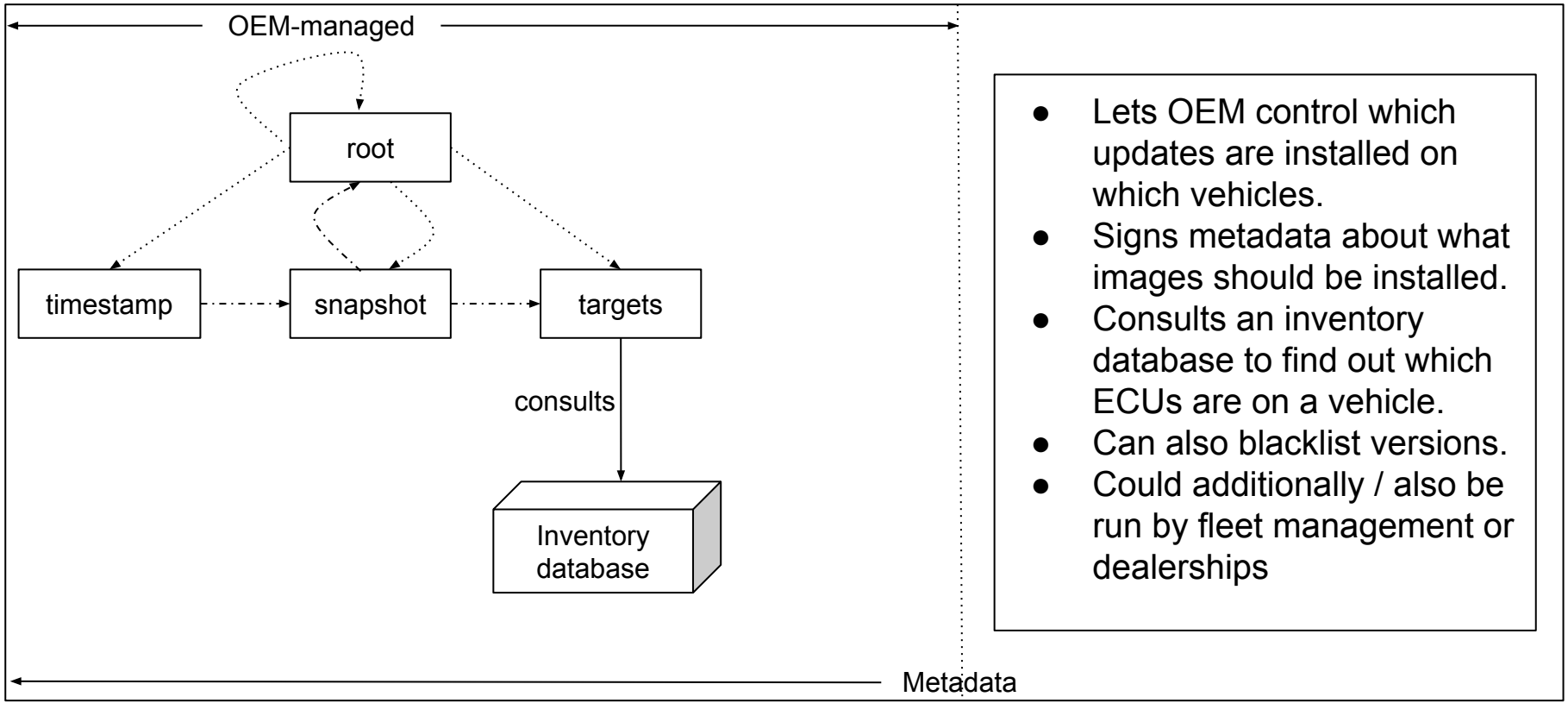
- When possible, OEM delegates updates for ECUs to suppliers.
- Delegations are flexible, and accommodate a variety of arrangements.

The image repository



- When possible, OEM delegates updates for ECUs to suppliers.
- Delegations are flexible, and accommodate a variety of arrangements.

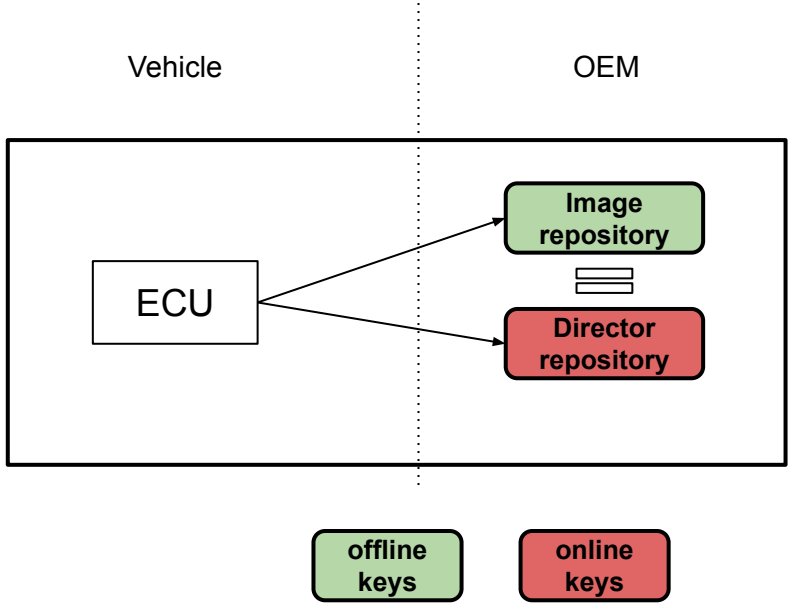
The director repository



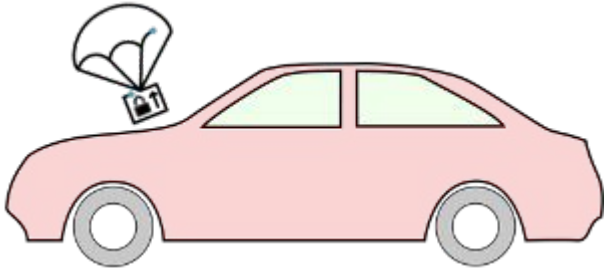
- Lets OEM control which updates are installed on which vehicles.
- Signs metadata about what images should be installed.
- Consults an inventory database to find out which ECUs are on a vehicle.
- Can also blacklist versions.
- Could additionally / also be run by fleet management or dealerships

Takeaway: security & flexibility

- Uptane provides both on-demand customization of vehicles & compromise-resilience.
- Gives an OEM a powerful array of options in controlling how updates are chosen for a vehicle, and who signs for updates.

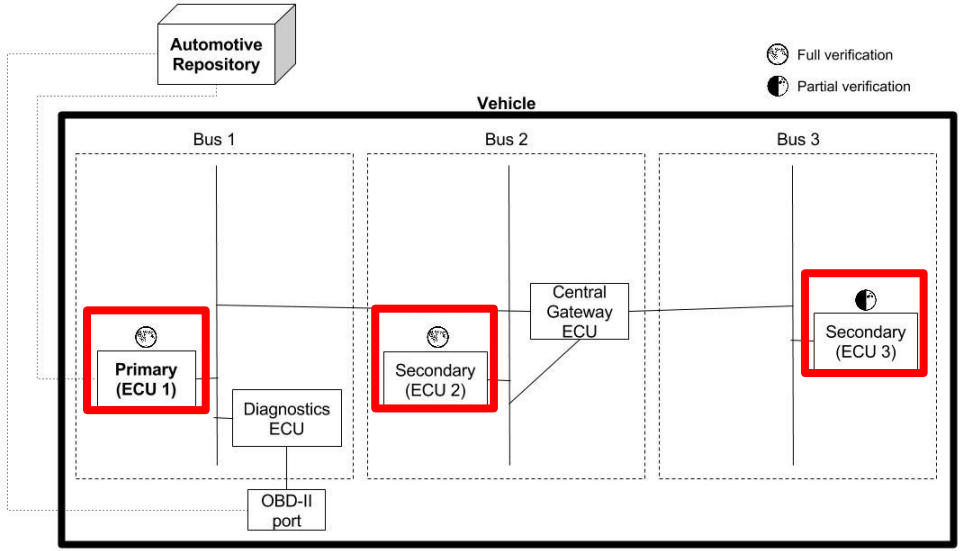


Verifying metadata & images on vehicles



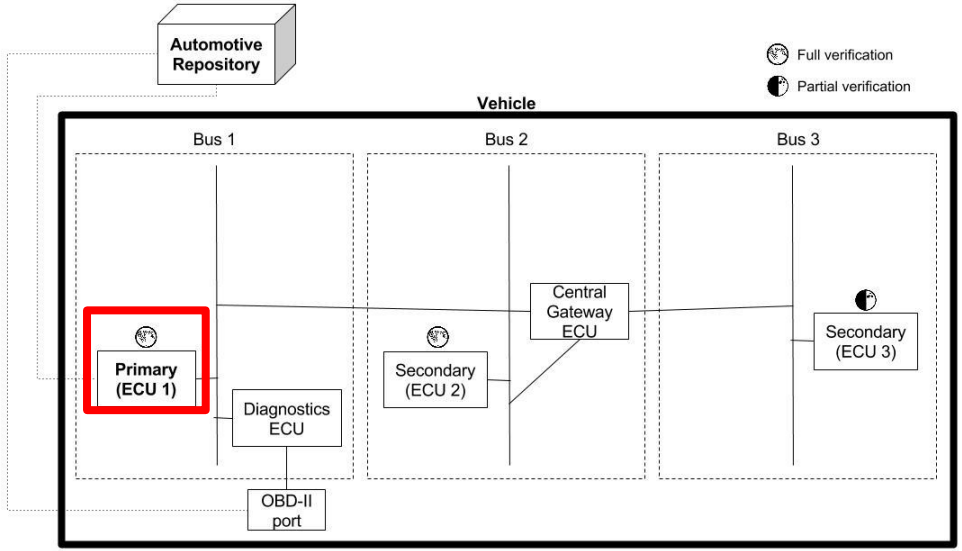
Primaries and secondaries

- Three types of ECUs, because:
  - Some ECUs are more / less powerful than others.
  - Few ECUs have network connection to outside world.
  - ECUs should not download metadata independently of each other.



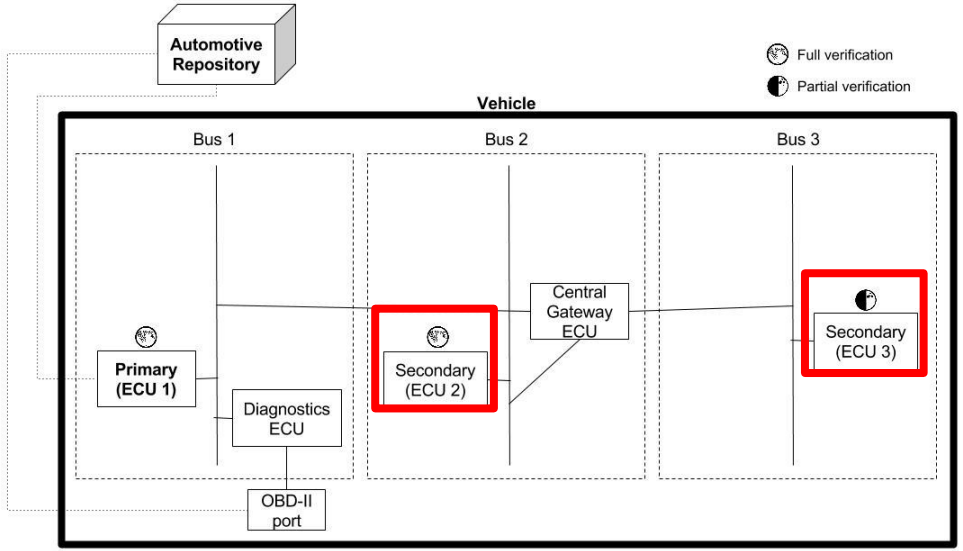
Primaries

- A primary downloads, verifies, distributes metadata + images to secondaries.



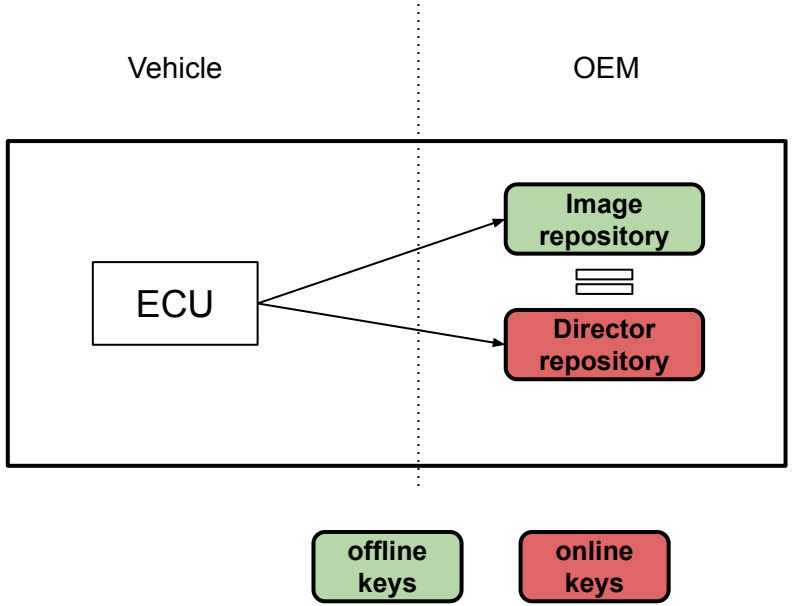
Partial / Full Verification Secondaries

- A secondary verifies both the metadata & image distributed by a primary, before updating to that image.



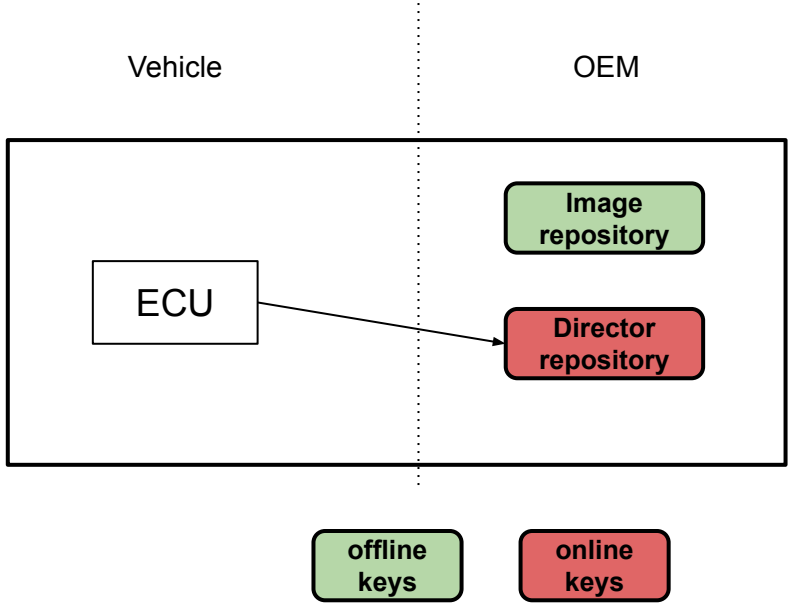
Full verification secondaries

- Checking that metadata about updates chosen by the director repository matches metadata about the same updates on the image repository.
- Involves checking ~3-6 signatures on metadata files

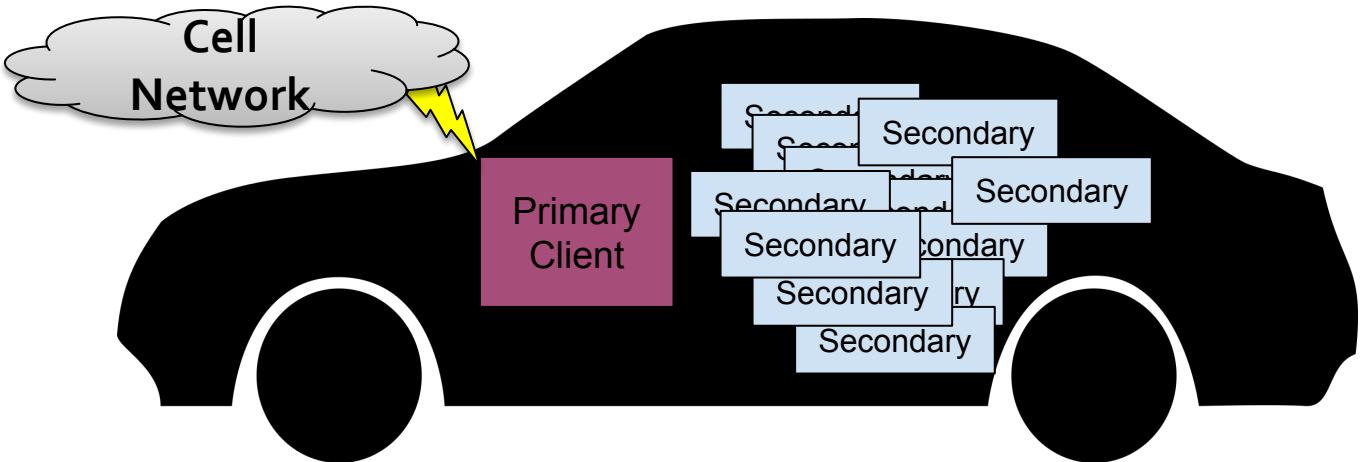


Partial verification secondaries

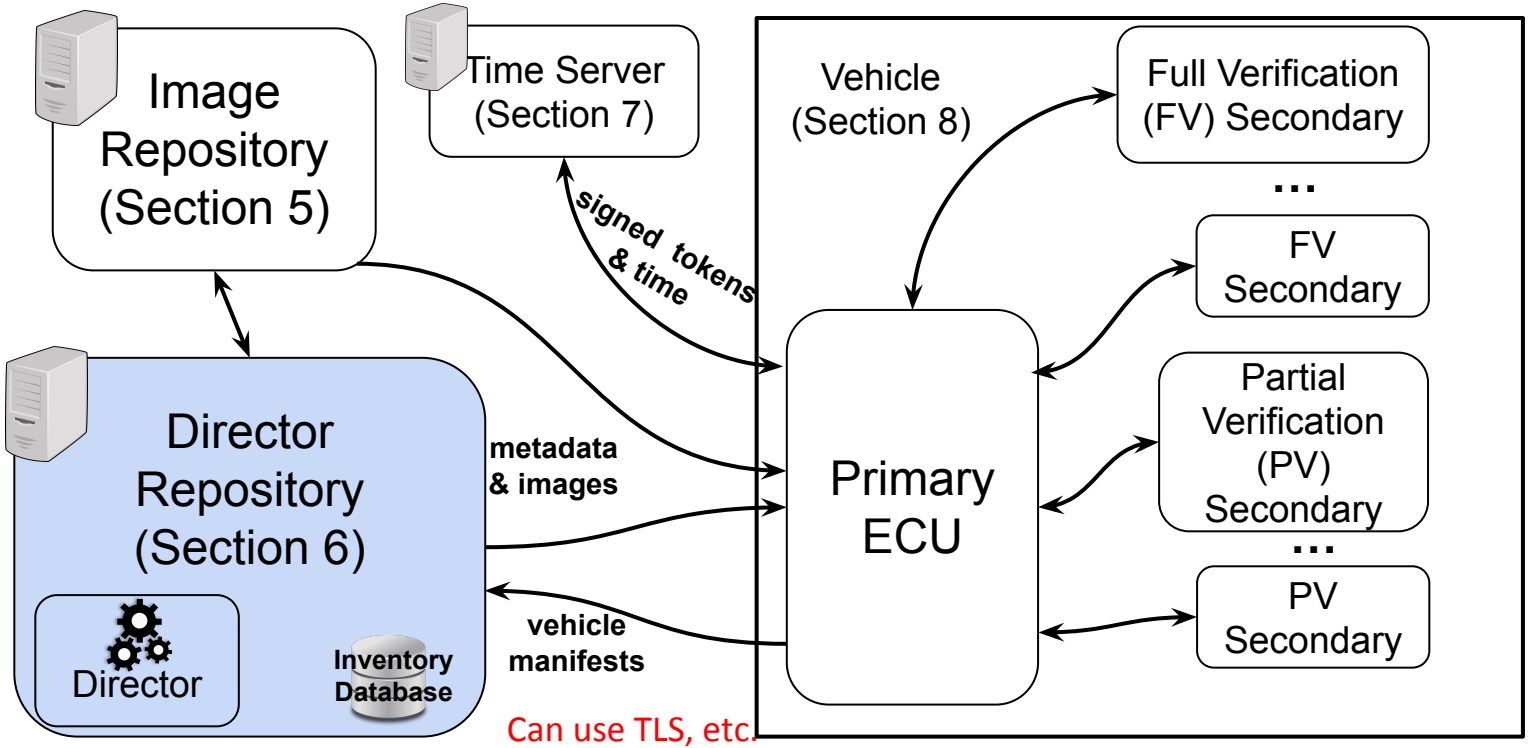
- Checking only metadata from the director repository.
- Involves checking only one signature on one metadata file.



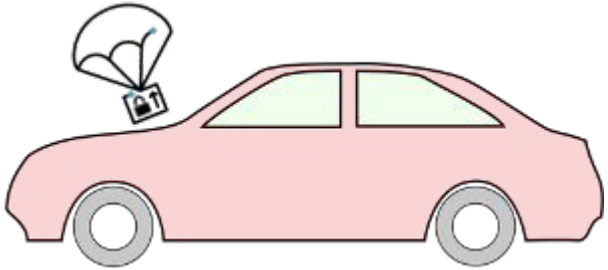
Uptane: Client-side Basics



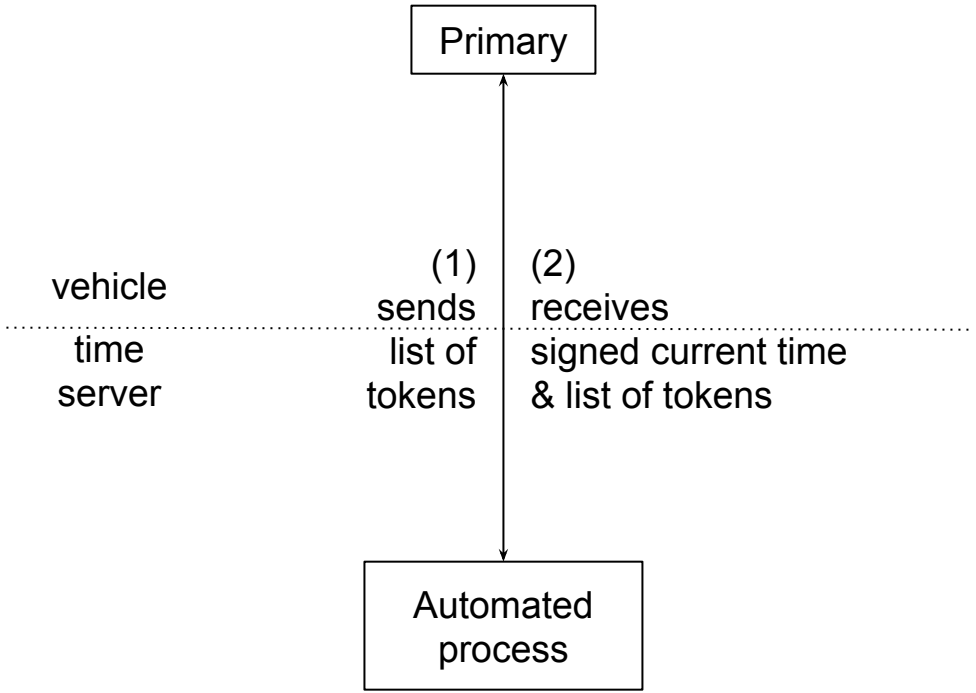
Uptane: High level view



Time server (optional)

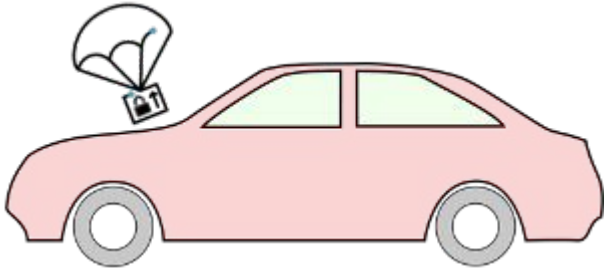


Time server (optional)

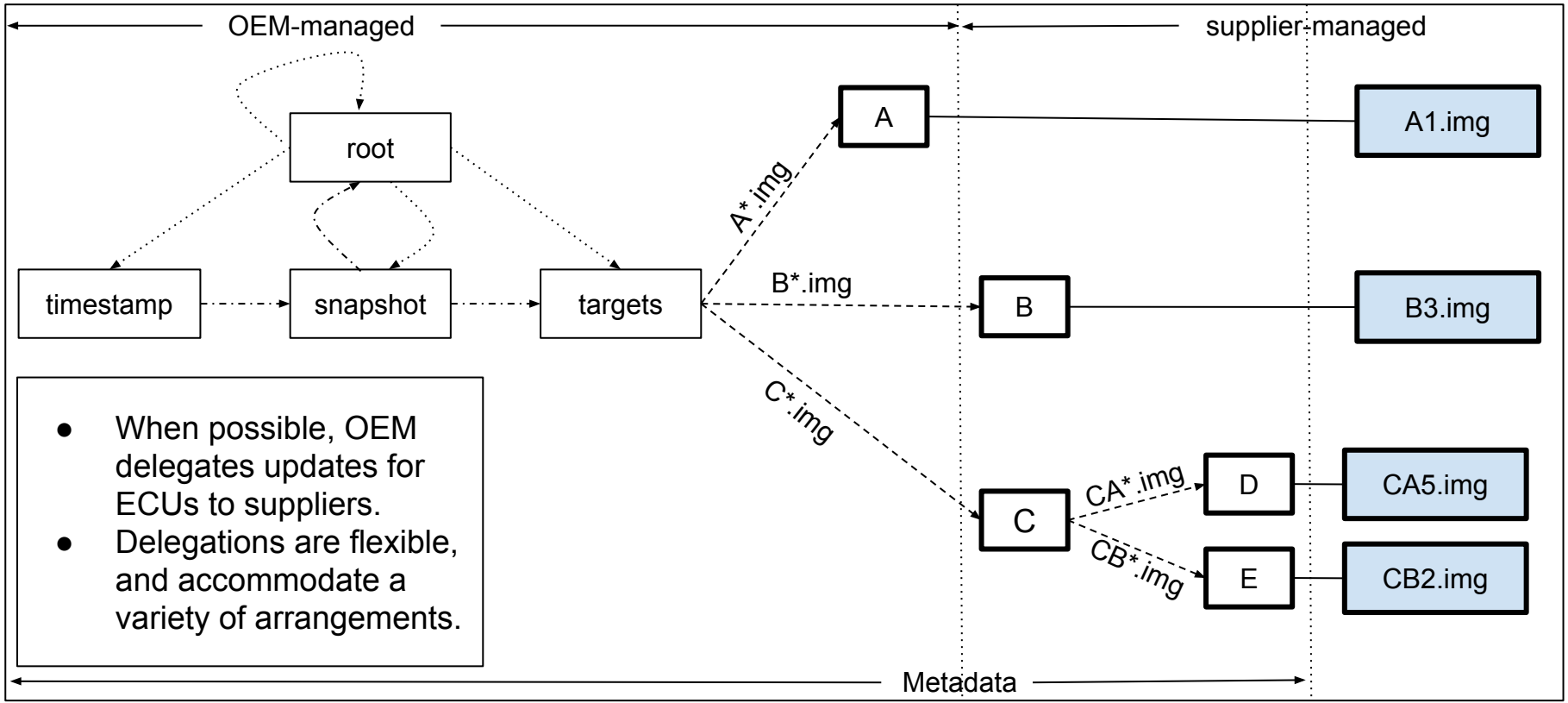


- A primary sends a list of tokens, one for each ECU, to the time server.
- An automated process on the time server returns a signed message containing: (1) the list of tokens, and (2) the current time.

Image repository

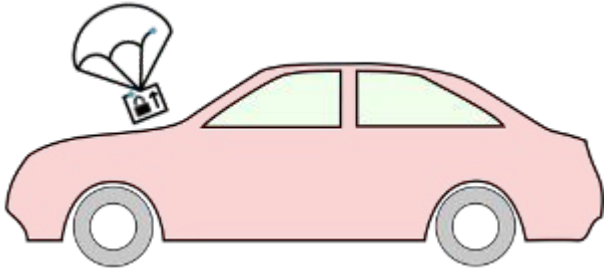


The image repository



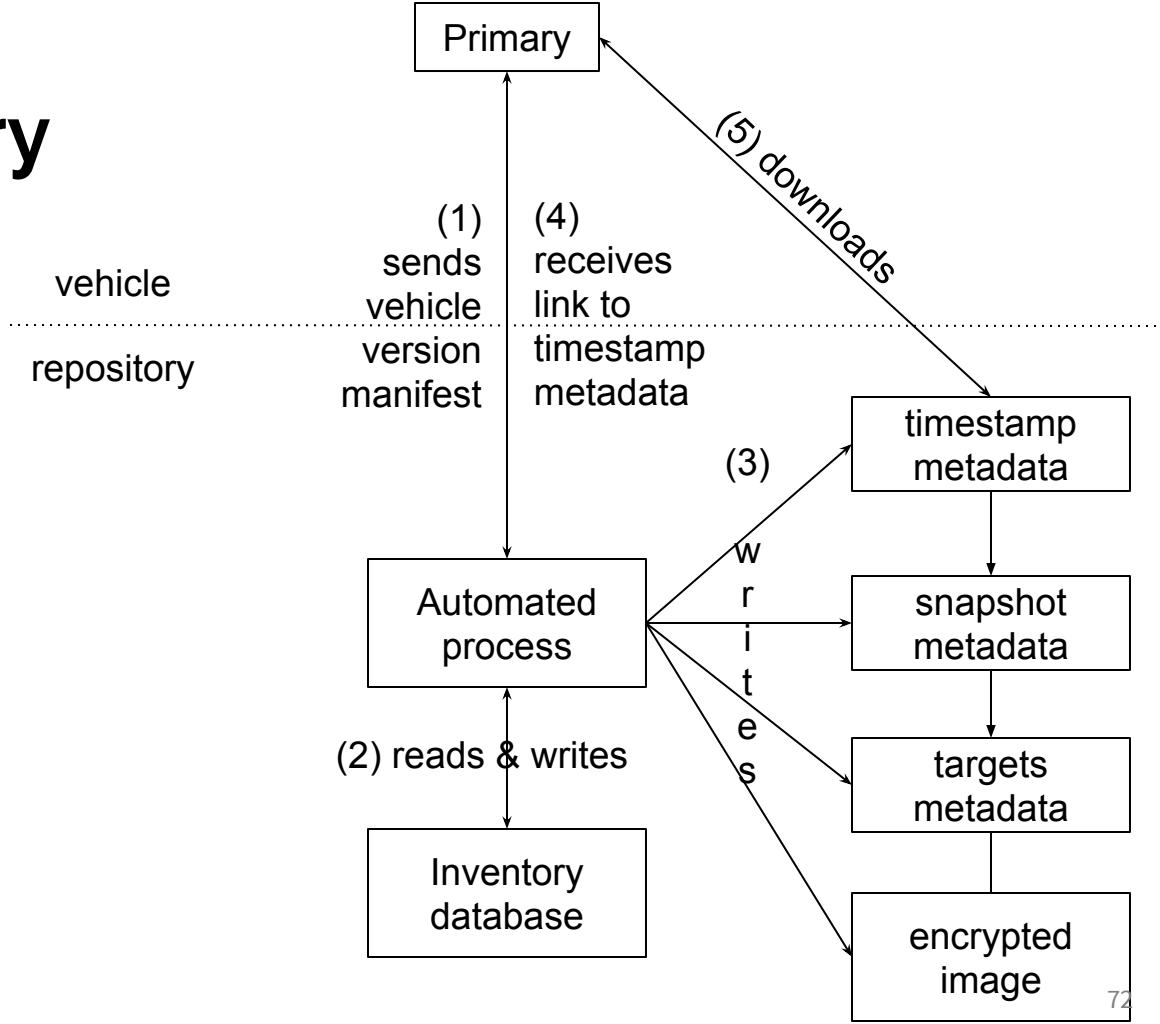
- When possible, OEM delegates updates for ECUs to suppliers.
- Delegations are flexible, and accommodate a variety of arrangements.

Director repository

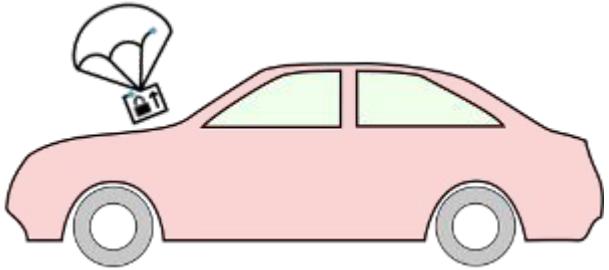


Director repository

- Records vehicle version manifests.
- Determines which ECUs install which images.
- Produces different metadata for different vehicles.
- May encrypt images per ECU.
- Has access to an inventory database.

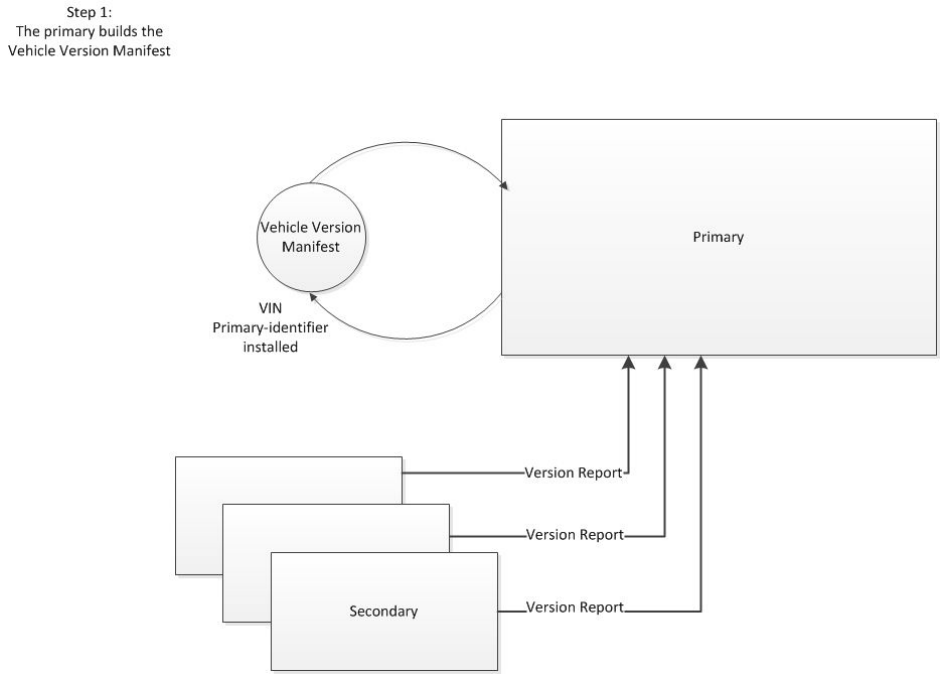


Uptane workflow on vehicle



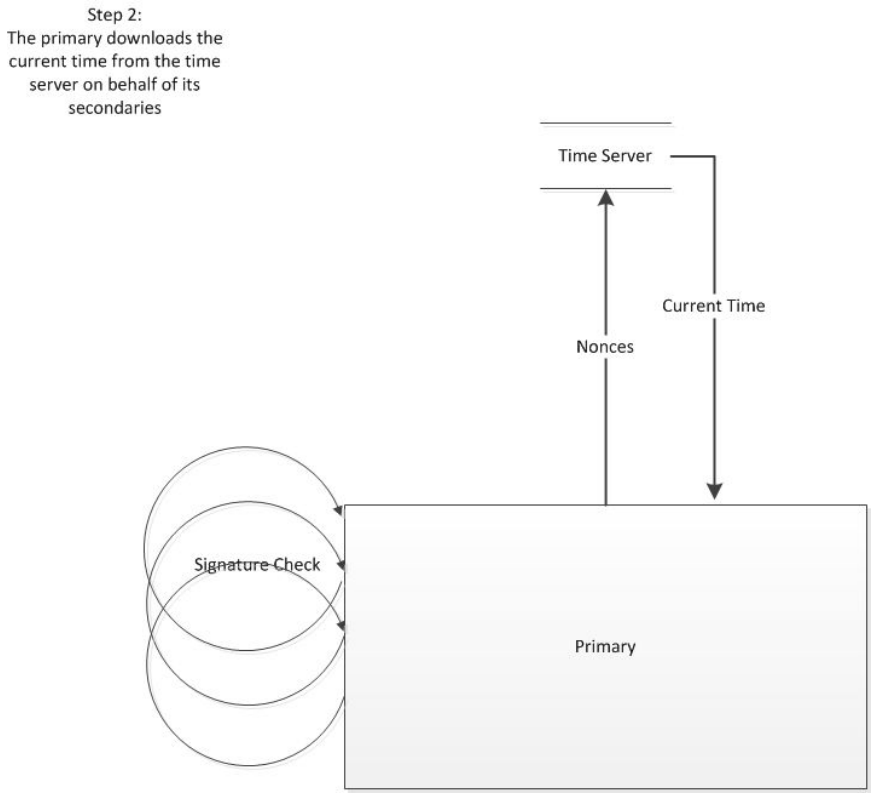
Downloading updates (1)

- Primary receives an ECU Version Manifest and a nonce from each Secondary.
- Primary produces Vehicle Version Manifest, a signed record of what is installed on Secondaries
- Primary sends VVM to Director
- Primary sends nonces to Timeserver



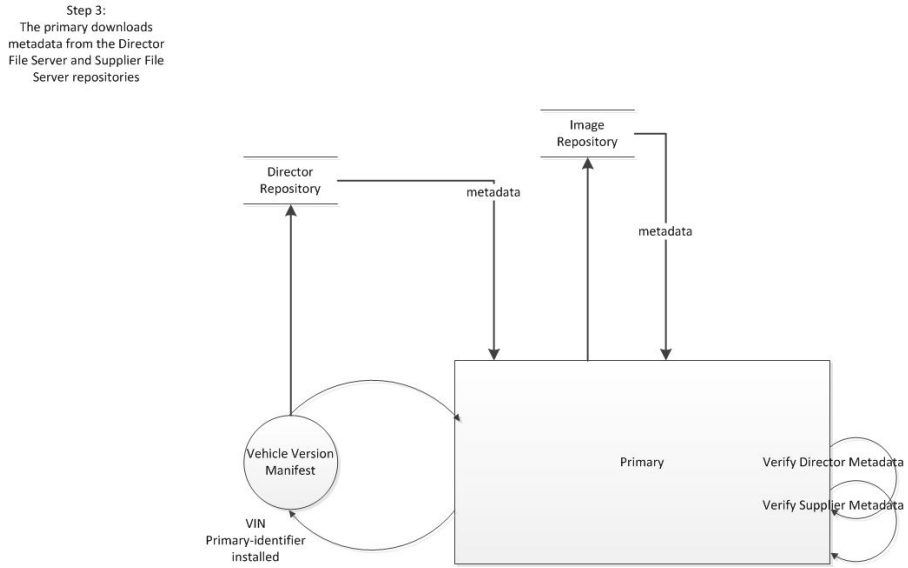
Downloading updates (2)

- Timeserver returns the signed [time and nonces] to the Primary.



Downloading updates (3)

- The primary downloads metadata from both the Director and Image repositories on behalf of all ECUs
- The primary performs *full verification* of metadata on behalf of all secondaries.



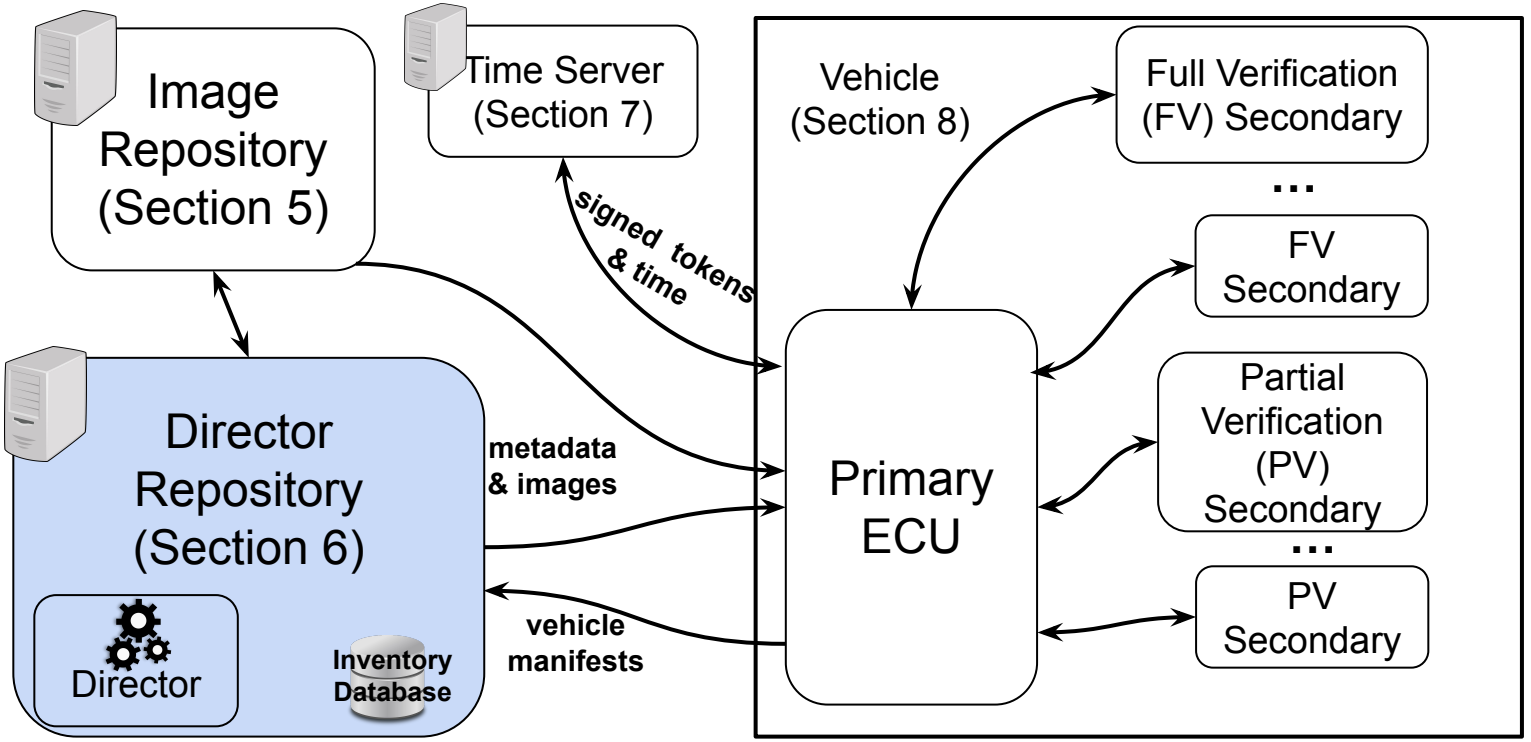
Full verification

1. **Load the latest downloaded time from the time server.**
2. **Verify metadata from the director repository.**
    a. Check the root metadata file.
    b. Check the timestamp metadata file.
    c. Check the snapshot metadata file.
    d. Check the targets metadata file.
3. **Download and verify metadata from the image repository.**
    a. Check the root metadata file.
    b. Check the timestamp metadata file.
    c. Check the snapshot metadata file, especially for rollback attacks.
    d. Check the targets metadata file.
    e. For every image A in the director targets metadata file, perform a preorder depth-first search for the same image B in the targets metadata from the image repository, and check that A = B.
4. **Return an error code indicating a security attack, if any.**

Partial verification

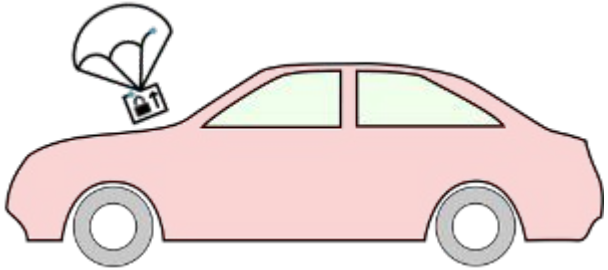
1. **Load the latest downloaded time from the time server.**
2. **Load the latest top-level targets metadata file from the director repository.**
    a. Check for an arbitrary software attack. This metadata file must have been signed by a threshold of keys specified in the previous root metadata file.
    b. Check for a rollback attack.
    c. Check for a freeze attack. The latest downloaded time should be < the expiration timestamp in this metadata file.
    d. Check that there are no delegations.
    e. Check that every ECU identifier has been represented at most once.
3. **Return an error code indicating a security attack, if any.**

Big picture



Can use TLS, etc.

Security properties

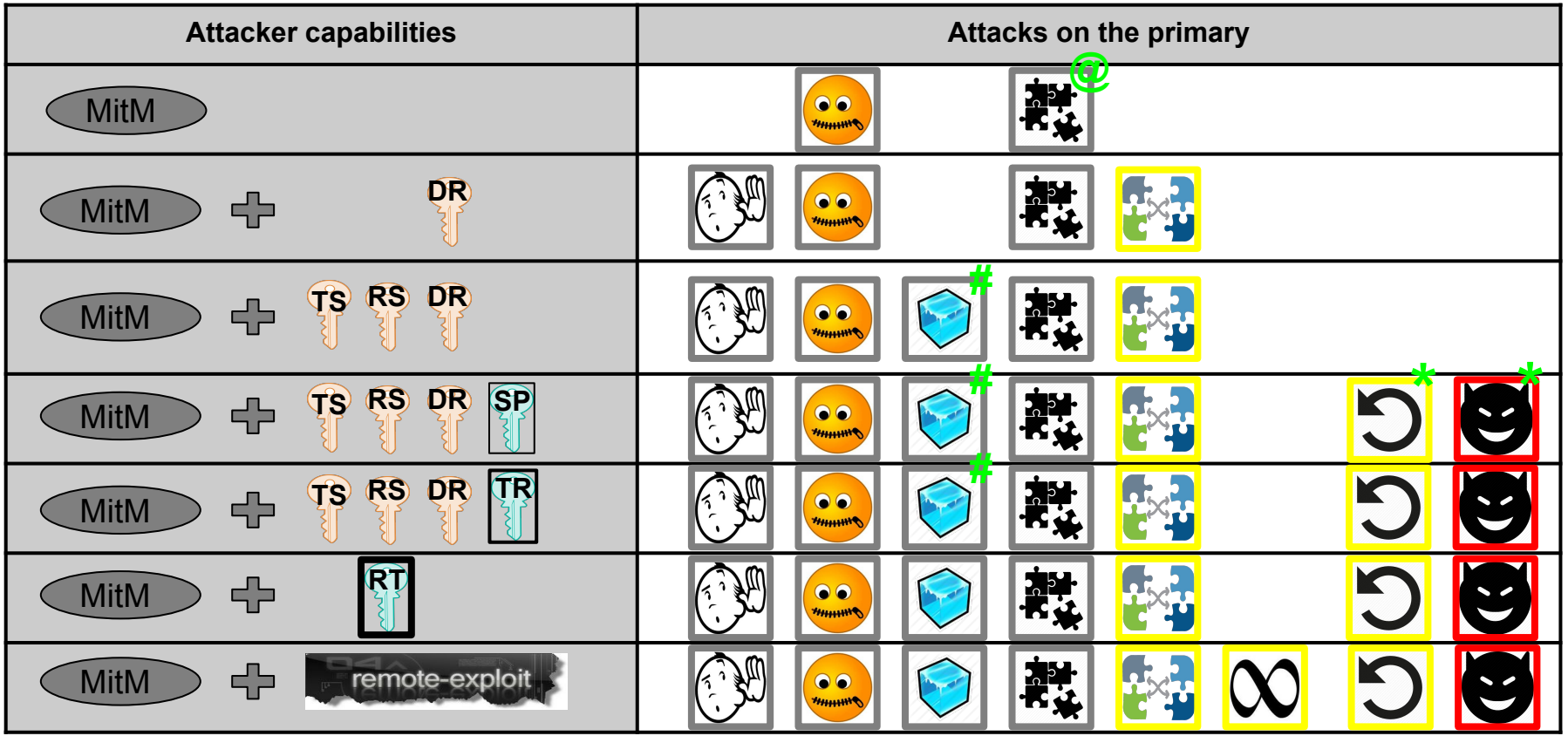


Optional security features

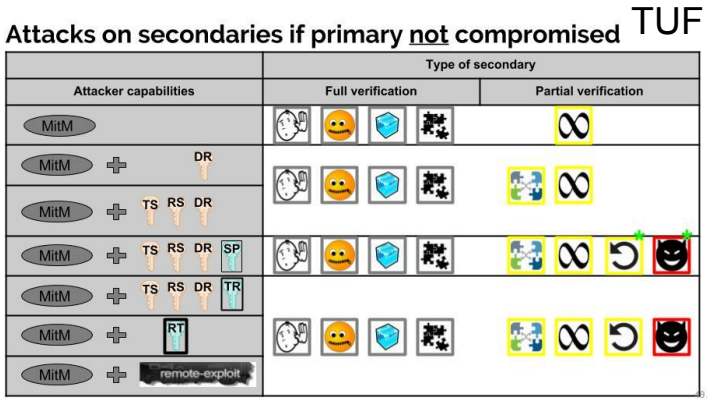
1. *Additional storage* to recover from endless data attacks 

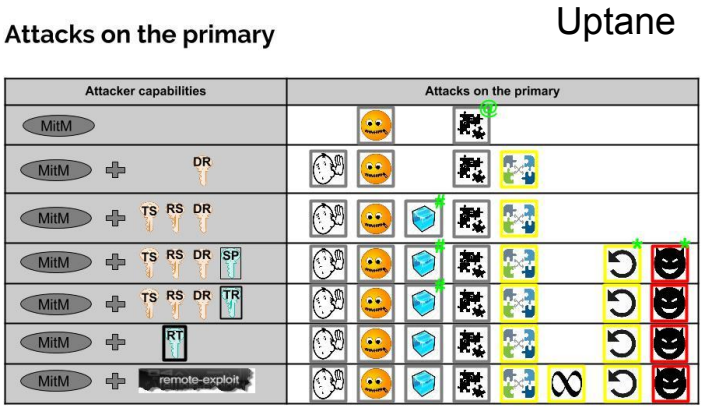
2. *Time server* to limit freeze attacks 

Attacks on the primary



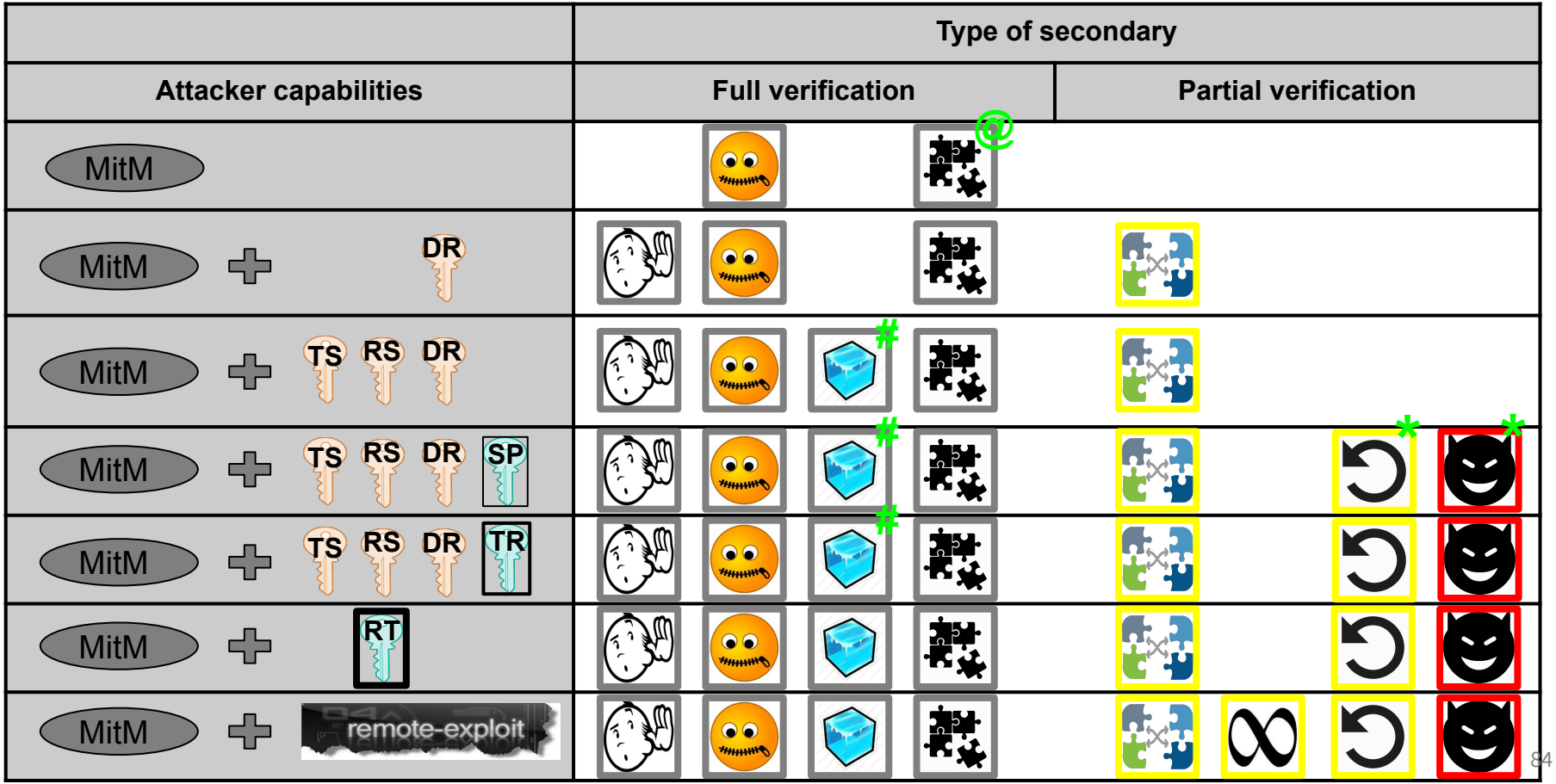
Attacks on the primary: comparison



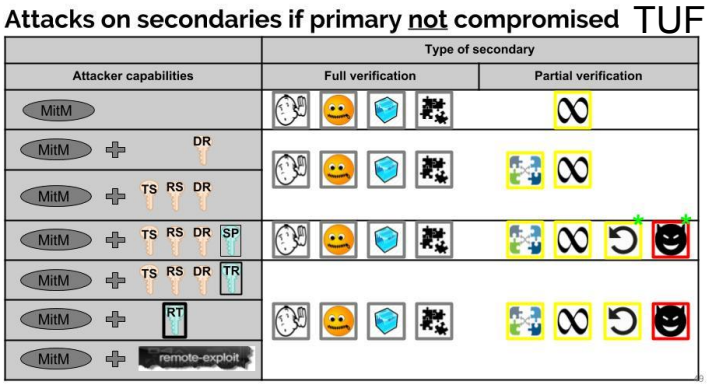


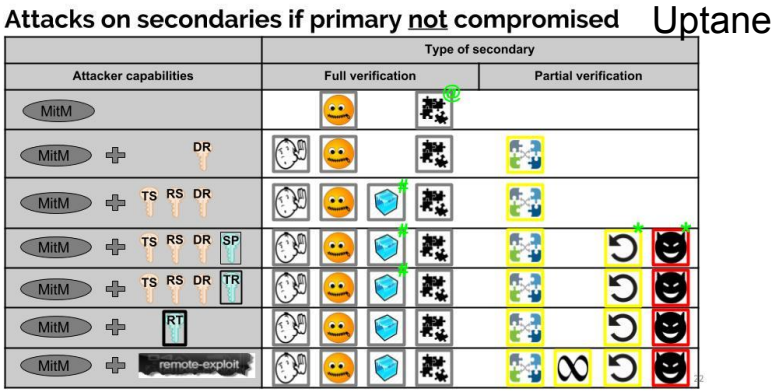
1. Eavesdrop attacks: not vulnerable when no director keys.
2. Partial bundle installation attacks: can be detected (and fixed) by director.
3. Freeze attacks: now needs timestamp, release, and director keys. Limited till earliest expiration timestamp.

Attacks on secondaries if primary not compromised



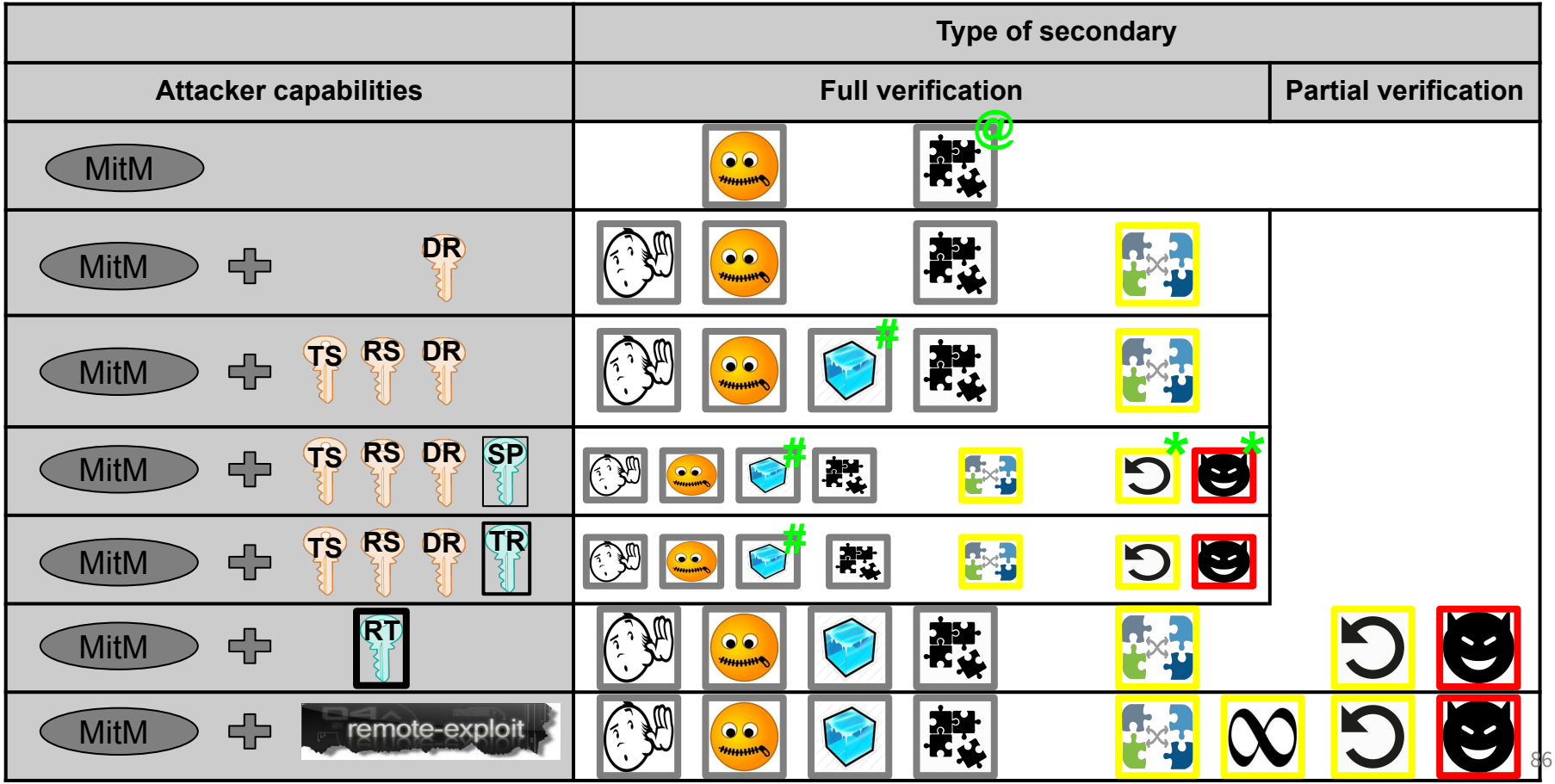
Attacks on secondaries: comparison





1. Endless data attacks: no secondary vulnerable (unless remotely exploited), because bootloader can restore from previous working image on additional storage.

Attacks on secondaries if primary compromised

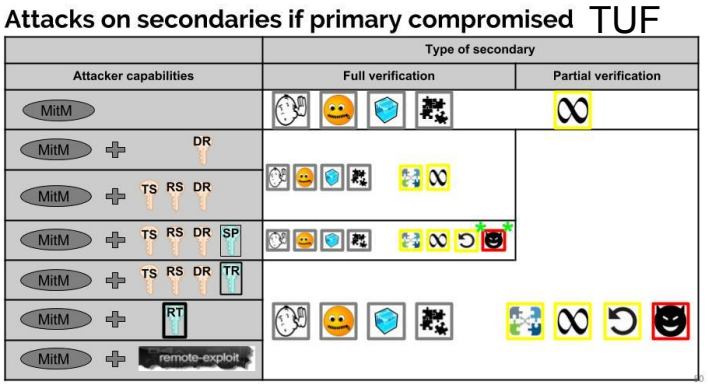


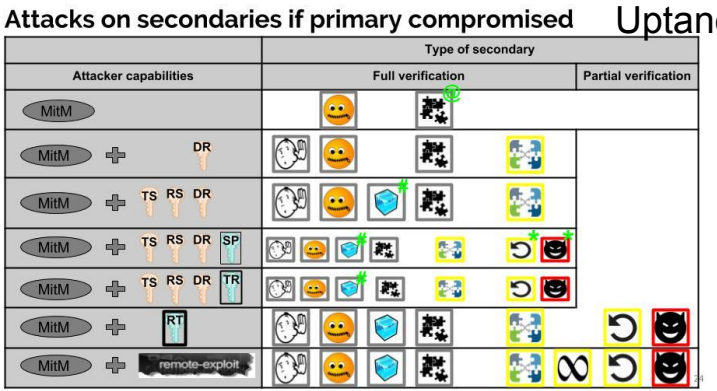
Attacks on secondaries if primary compromised

Attacker capabilities	Type of secondary: Full verification	Type of secondary: Partial verification
MitM		
MitM +		
MitM + T		
MitM + TS RS DR SP		
MitM + TS RS DR TR		
MitM + RT		
MitM + remote-exploit		

OMA-DM, ITU-T X.1373, etc. enable full control with a single compromise

Attacks on secondaries if primary compromised: comparison



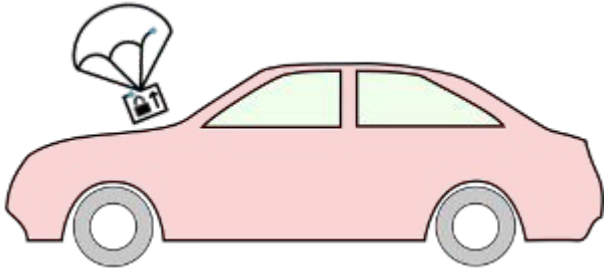


- Differences from when primary not compromised
  - When director keys are compromised, rollback & arbitrary software attacks on ALL partial verification secondaries on ALL vehicles.
  - Full verification secondaries NOT affected until at least the right supplier's keys are compromised.

Attacks on secondaries if primary compromised: comparison

- Differences from when primary not compromised
  - When director keys are compromised, rollback & arbitrary software attacks on ALL partial verification secondaries on ALL vehicles.
  - Full verification secondaries NOT affected until at least the right supplier's keys are compromised.

Deployment



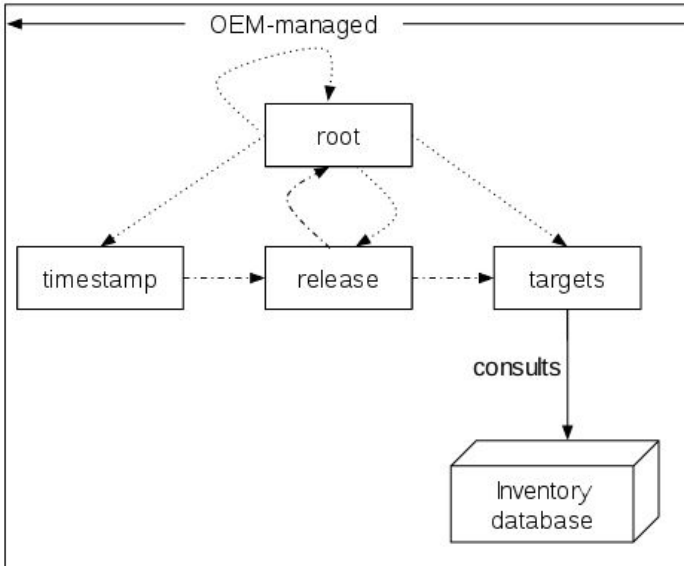
What changes are needed to use Uptane?

1. OEM sets up and maintains
   - Director repository
   - Image repository
   - Time server (optional)
2. Images are signed by
   - Supplier, or
   - OEM, or
   - Both!
3. ECUs shall do either
   - Full verification, or
   - Partial verification
4. May keep using your existing TLS, etc. transport
   - If transport / caching compromised, little security risk

In practice OEMs have these pieces already...

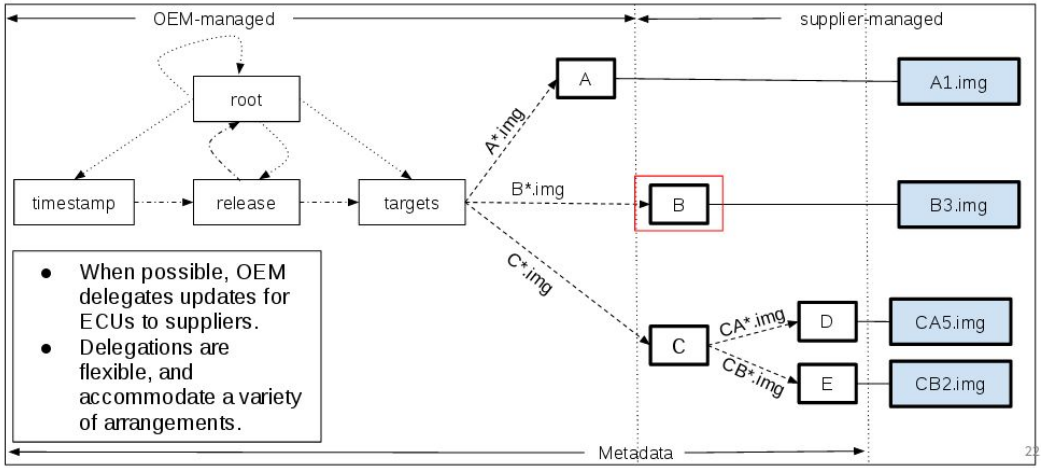
OEM: director repository

- Used to instantly respond to new information
  - Typically used to instruct a vehicle which updates to install, depending on what it has
  - Can be used to instantly blacklist updates
- Wholly automated
  - Online keys
  - Use Uptane API to generate signed metadata
  - Uses an inventory database to read and write information about ECUs (e.g., public keys, what was previously installed, etc.)



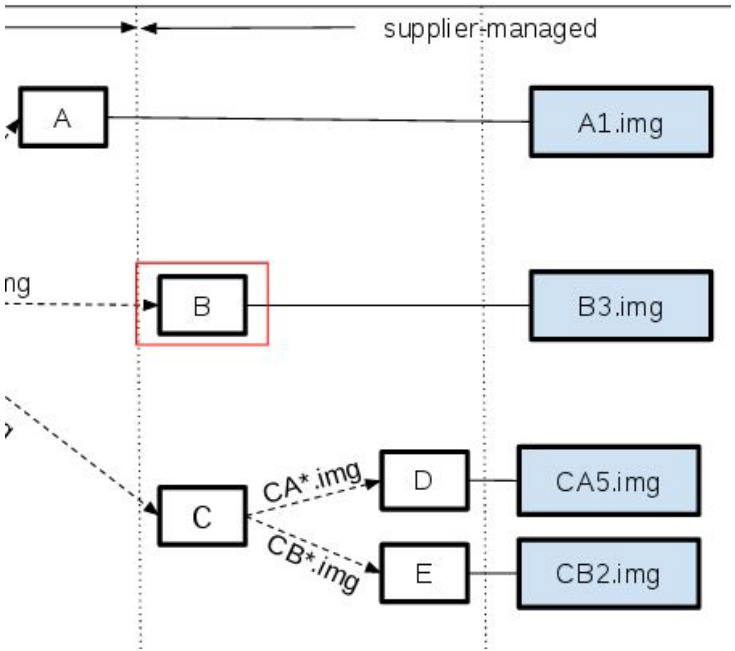
OEM: image repository

- Used to publish images produced by suppliers
- Occasional administration
  - Periodically (e.g., weekly, monthly) update metadata about available images
  - Use Uptane command-line tools to generate metadata
  - Use threshold of offline keys (e.g., Yubikey, HSM, etc. often used) to sign metadata



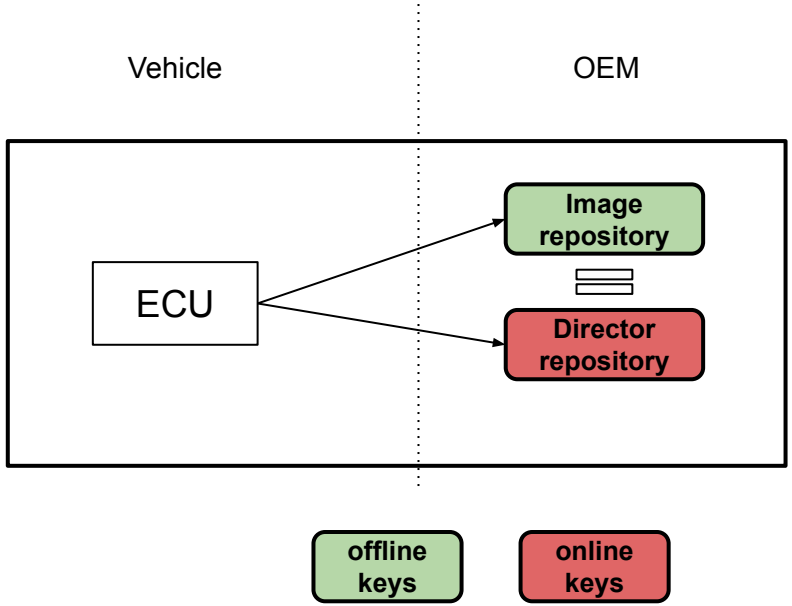
Supplier

- Supplier should sign metadata about images
  - Run a single command to produce metadata
  - Keys must be offline for security
  - Could further delegate to teams / suppliers
  - Used when producing a new image for deployment
  - Could use a threshold of keys if they elect
- Upload metadata and images to OEM
- May be done by OEM on behalf of supplier

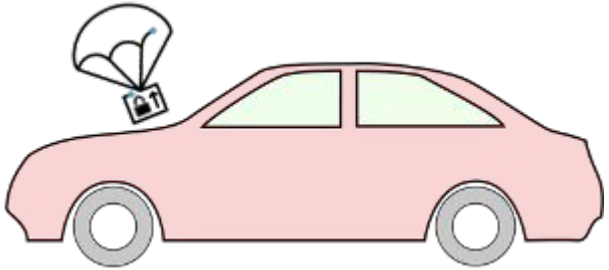


ECU

- Full verification
  - For safety-critical ECUs that should not be hacked
  - Optionally, use additional storage space to be able to rollback in case of emergency
- Partial verification
  - For ECUs with speed and / or memory constraints
  - If cannot do this, then do not update OTA!
- Each ECU should store one key
  - Asymmetric key preferred, but not required



Uptane status / wrap up



Uptane an Open and Secure SOTA system

- Multiple open source, free to use implementations
  - C++ (Automotive Grade Linux), C, Python reference implementation

- Diverse set of vendors and integrators
    - Robust participation from dozens of organizations (vendors, OEMs, regulators, security experts, etc.)
  - Solid, battle-tested technology mandated by several OEMs
  - Completely free / no license or patent restrictions
  - We welcome other interested parties to participate

- Uptane meets and surpasses existing regulatory proposals for security
  - Tech based upon widely deployed, advanced security systems
  - **Upcoming regulation is mandating compromise resilience**

Uptane Standardization

- Open, Community standardization effort
  - Completely free to join
    - All funding from DHS (US Government), no vendor / OEM payment needed
  - IEEE / ISTO standard (1.0.0)
  - Linux Foundation JDF project
    - Future revisions: ISO standardization
  - Testing Plan and Deployment Considerations standardization in progress
  - All documents are open and free to use

Security Reviews

Reviews of implementations and design:

- Cure53 audited ATS's Uptane implementation
- NCC Group audited Uptane's reference implementation (pre-TUF fork)
- SWRI provided Uptane reference implementation / specification audit
- ...

Uptane Integration

Work closely with vendors, OEMs, etc.

- Many top suppliers / vendors adopted Uptane in future cars!
    - About 1/3 new cars on US roads
- Automotive Grade Linux
- OEM integrations
    - Easy to integrate!

AUTOMOTIVE GRADE LINUX

in-toto secures the complete software supply chain!

Uptane integrates with in-toto

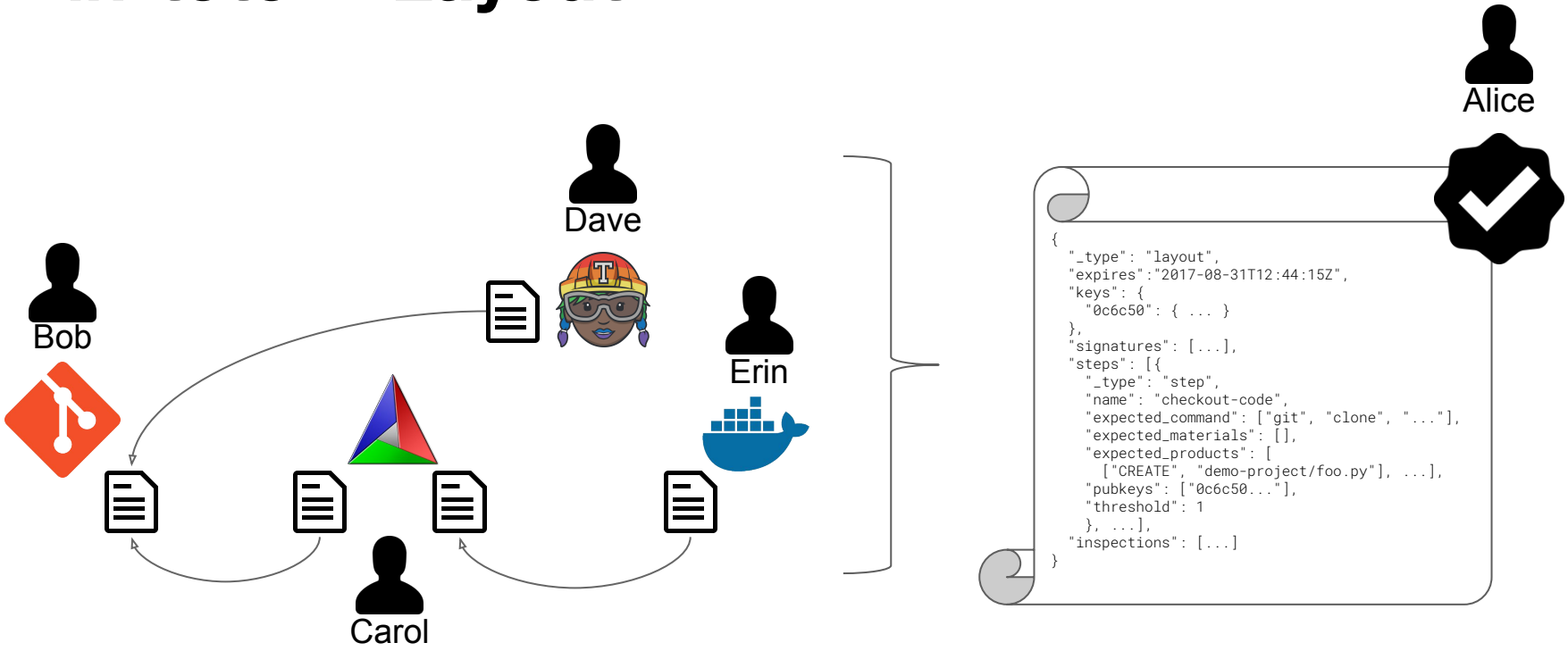
→ Verifiably define the steps of the software supply chain

→ Verifiably define the authorized actors

→ Guarantee that everything happens according to definition, and nothing else

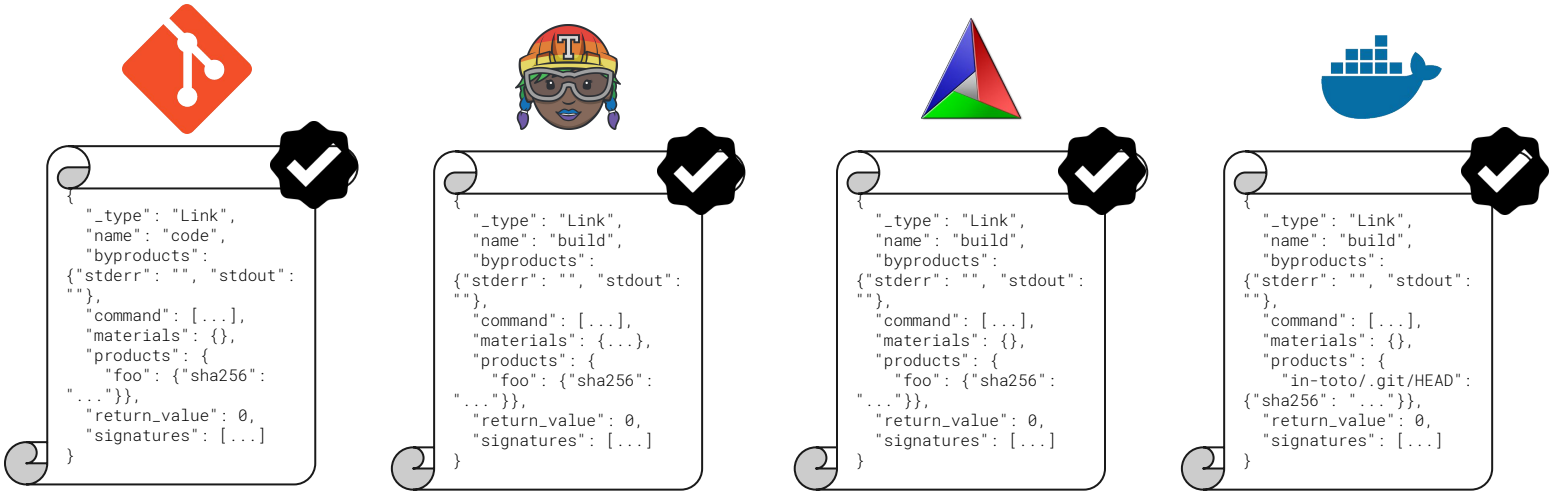
Sort of like Uptane for the supply chain

in-toto -- Layout



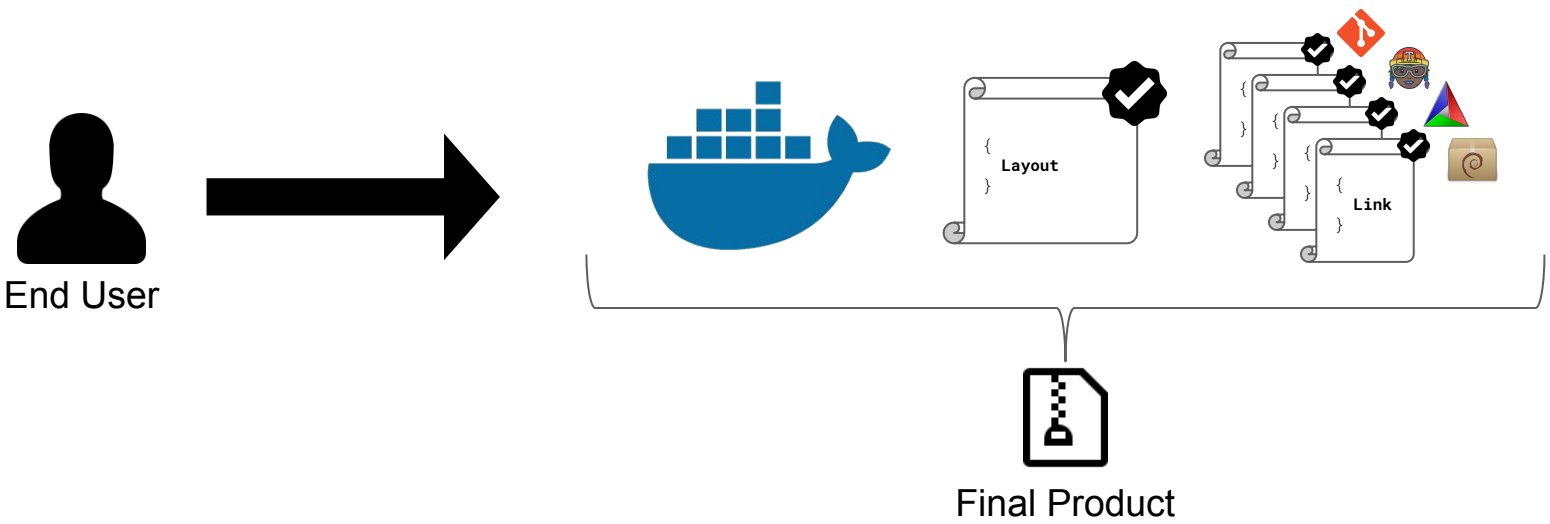
in-toto -- Link -- Attestation for each step

```
$ in-toto-run -- ./do-the-supply-chain-step
```



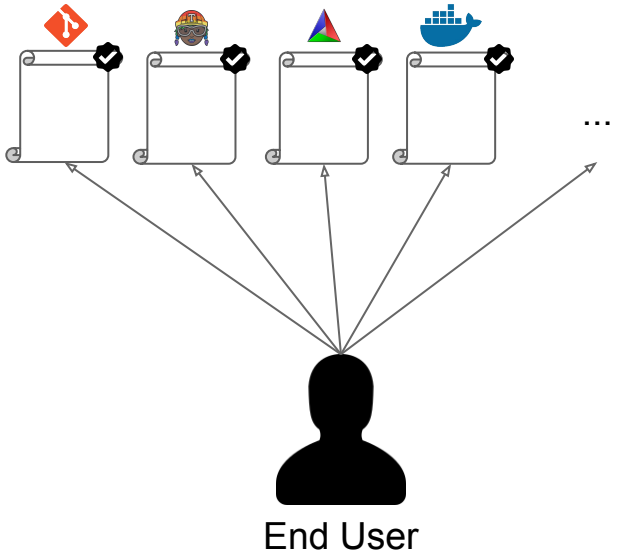
in-toto -- Verification

```
$ in-toto-verify --layout <layout> --key <pub key>
```



in-toto -- Inspections

- Used to verify metadata from within a step
- Performed by the client
- Uses link + additional (app specific) metadata and the layout

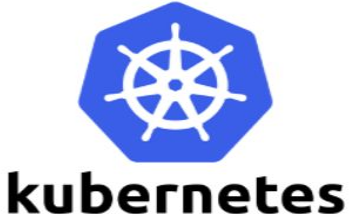


in-toto + Uptane

- in-toto cryptographically secures the whole supply chain
  - all the way right and left
  - Security grounding / principles from TUF
  - Prevents, detects, and **mitigates** compromises

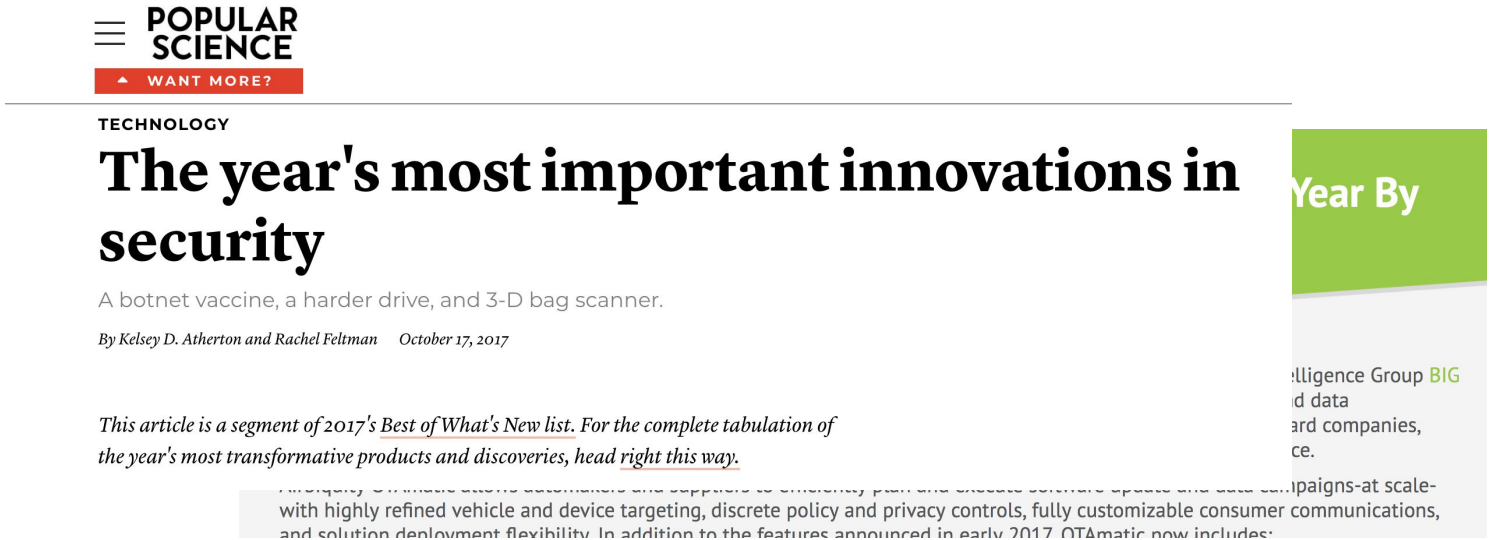
- Lots of production use

- Try out in-toto!
  - https://in-toto.io



Uptane Press

- Dozens of articles
- TV / Radio / Newspapers / Magazines



POPULAR SCIENCE

WANT MORE?

TECHNOLOGY

The year's most important innovations in security

A botnet vaccine, a harder drive, and 3-D bag scanner.

By Kelsey D. Atherton and Rachel Feltman October 17, 2017

This article is a segment of 2017's Best of What's New list. For the complete tabulation of the year's most transformative products and discoveries, head right this way.

Year By

lligence Group BIG
d data
ard companies,
ce.

paigns-at scale-
with highly refined vehicle and device targeting, discrete policy and privacy controls, fully customizable consumer communications,
and solution deployment flexibility. In addition to the features announced in early 2017, OTAmatic now includes:

What we want to avoid

- Some groups will elect to use insecure designs
  - Computer security designs are open / publicly reviewed for a reason!
    - Equivalent: Use SnakeOil proprietary brand symmetric encryption instead of AES, we have 7 more S-boxes!
    - Equivalent: Use SnakeOil proprietary brand crypto instead of TLS, we use less bandwidth and have a better slogan!
  - Don't fall for marketing tricks!
- Companies that do not secure their cars put lives at risk
  - Attacks will happen
  - Lawsuits will cost hundreds of millions of USD
    - Hiding behind weak regulation will not be effective

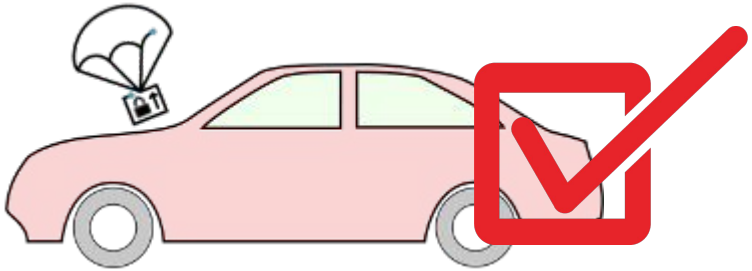
People will die!

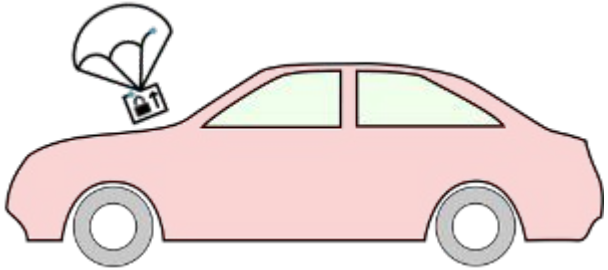
Get Involved With Uptane!

- Workshops
- Technology demonstration
- Compliance tests
- Standardization (IEEE / ISTO)
- Join our community! (email: jcappos@nyu.edu or go to the Uptane forum)

https://uptane.github.io/

Homeland Security





For more details, please see the Implementation Specification and other documentation at **uptane.github.io**