Uptane

Securing Over-the-Air Updates
Against Nation State Actors

Marina Moore
New York University

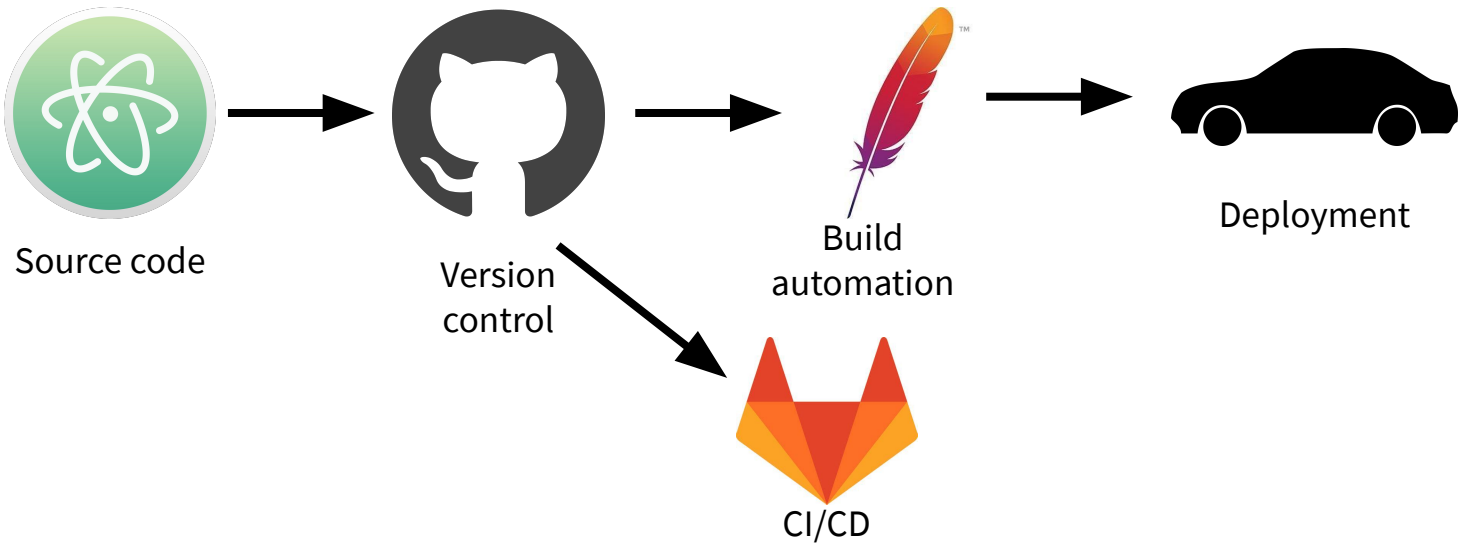
Ira McDonald

uptane.github.io

NYU | TANDON SCHOOL OF ENGINEERING

Software supply chain

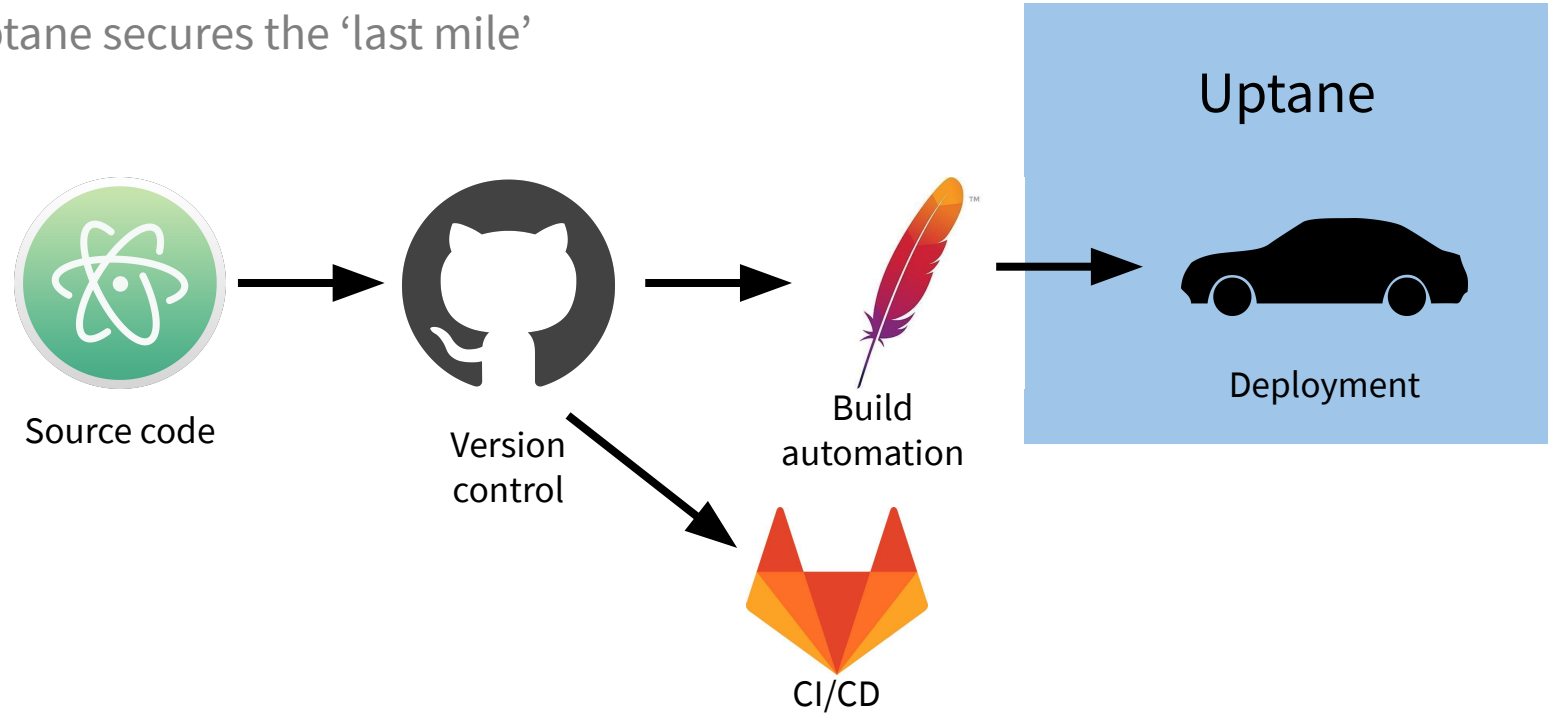
From developer's mind to your automobile



https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

Software supply chain

Uptane secures the ‘last mile’

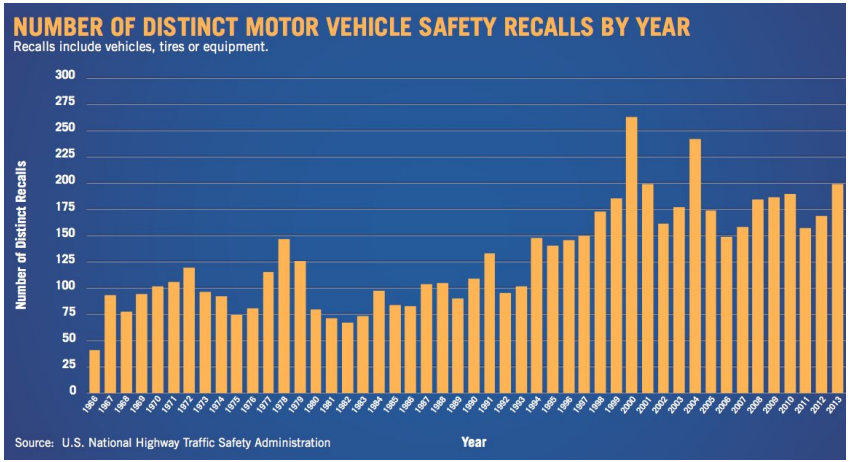


Why update software in cars

- 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 need to be over-the-air

- 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

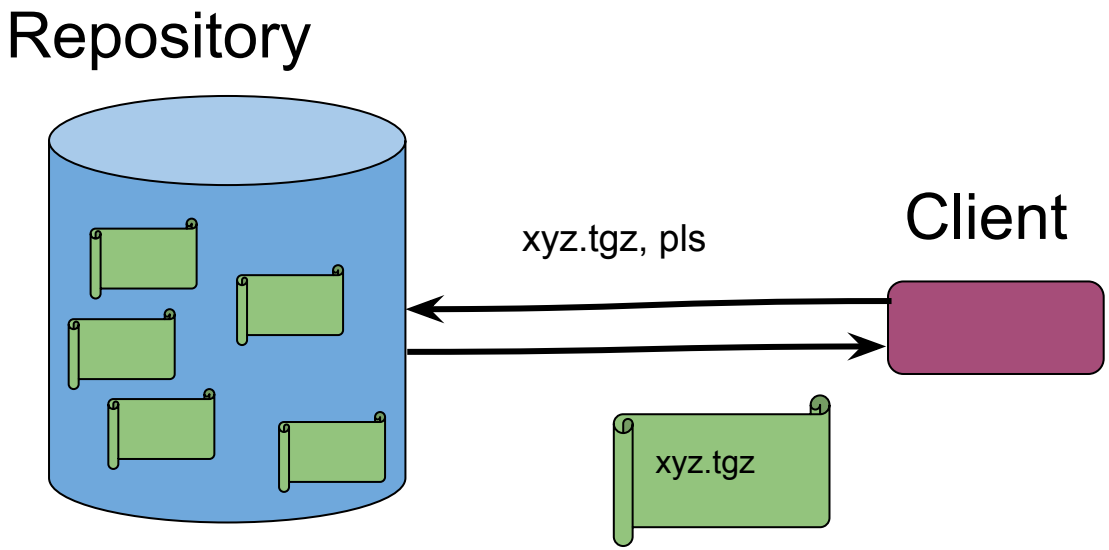
- Update -> Control
- Nation-state actors pull off complex supply chain attacks
    - Must not have a single point of failure

Compromises happen

- Sunburst attack on Solarwinds distributed by software update
- 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

RubyGems

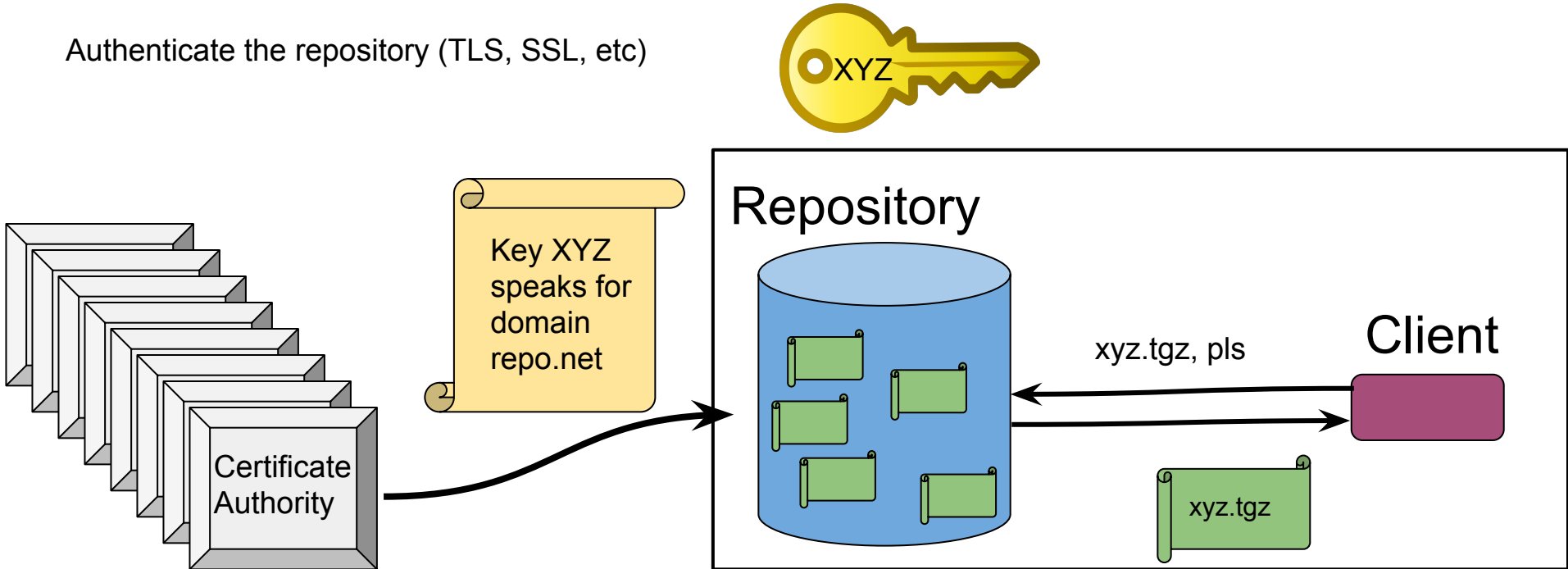
Update basics



Inadequate update security 1: TLS/SSL

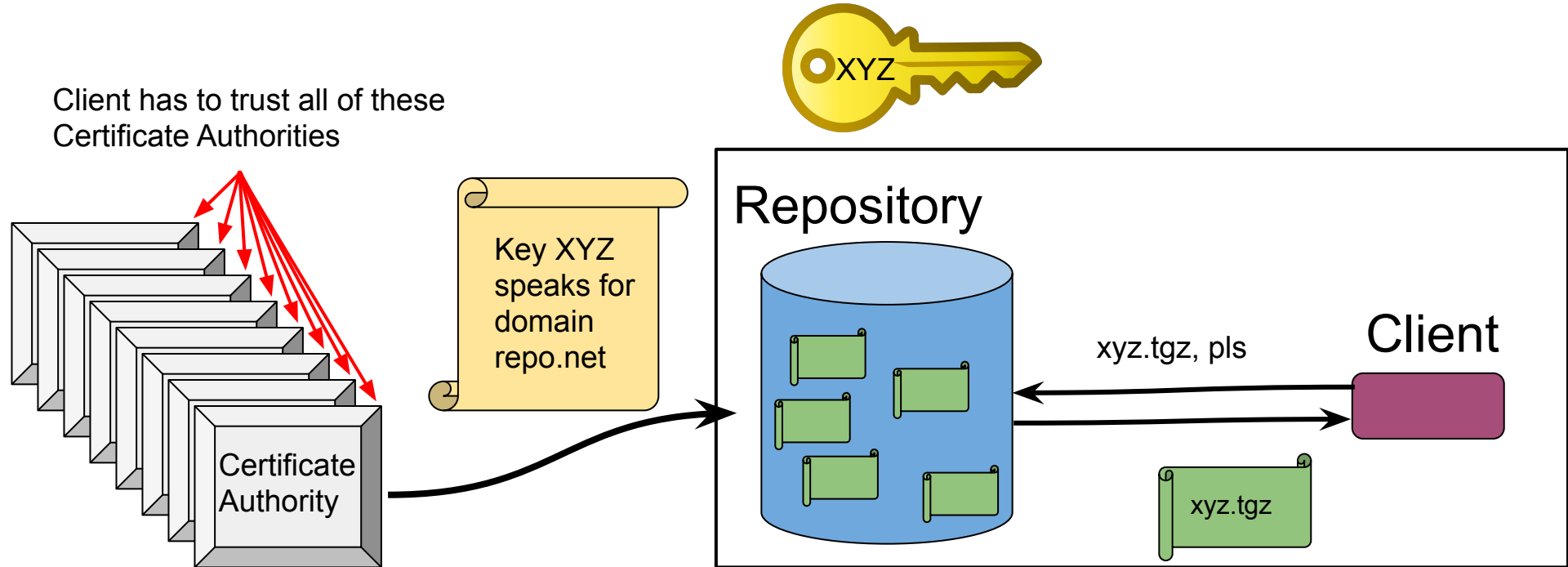
Traditional solution 1:

Authenticate the repository (TLS, SSL, etc)



Inadequate update security 2: TLS/SSL

Transport Layer Security: Problem 1

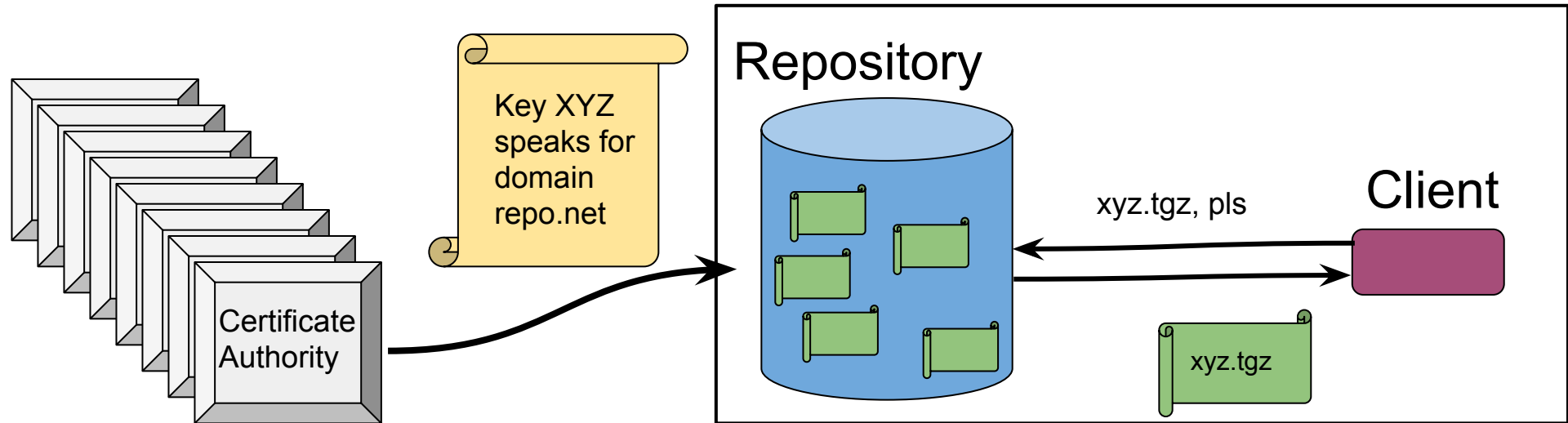


Inadequate update security 3: TLS/SSL

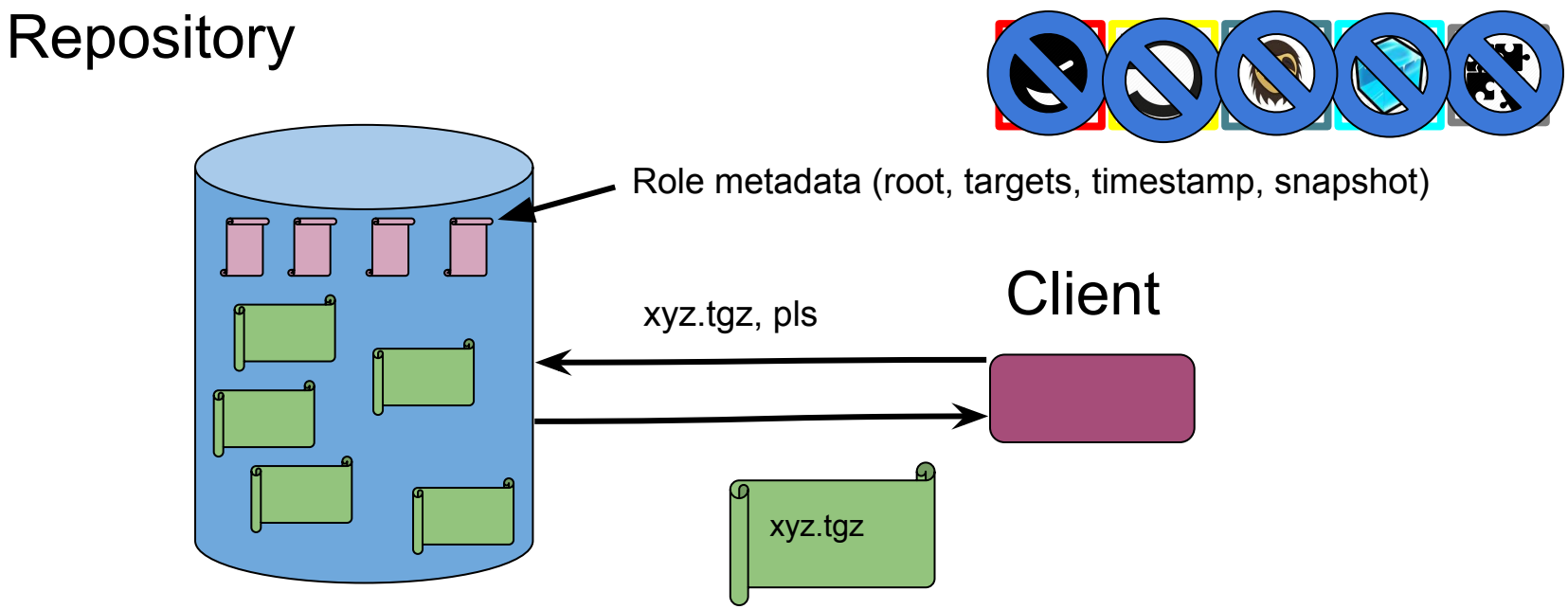
Transport Layer Security: Problem 2

Client has to trust this key.

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

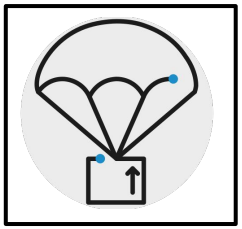


The Update Framework (TUF)



https://theupdateframework.io/

The Update Framework (TUF)



DATADOG

notary

DigitalOcean

vmware

IBM

Microsoft Azure

docker

QUAY by CoreOS

CLOUDFLARE

Google

amazon

Flynn

Uptane automotive SOTA goals

- Inspired by The Update Framework (TUF), a software update framework that is designed for compromise resilience and security
- Compromise resilient software-over-the-air (SOTA) updates
- Ensures that images on the repository match images the vehicle installs
- Avoid arbitrary package installation even when server is compromised
- Minimize damage from a compromised signing key
- Built-in key revocation
- Prevent known attacks on software update systems

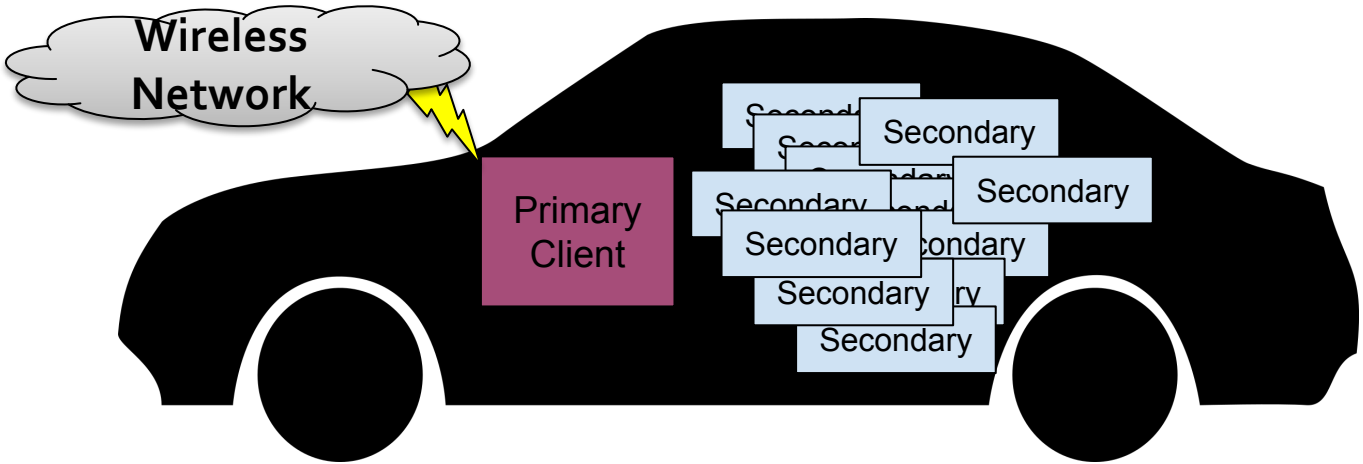
https://uptane.github.io/

Uptane design

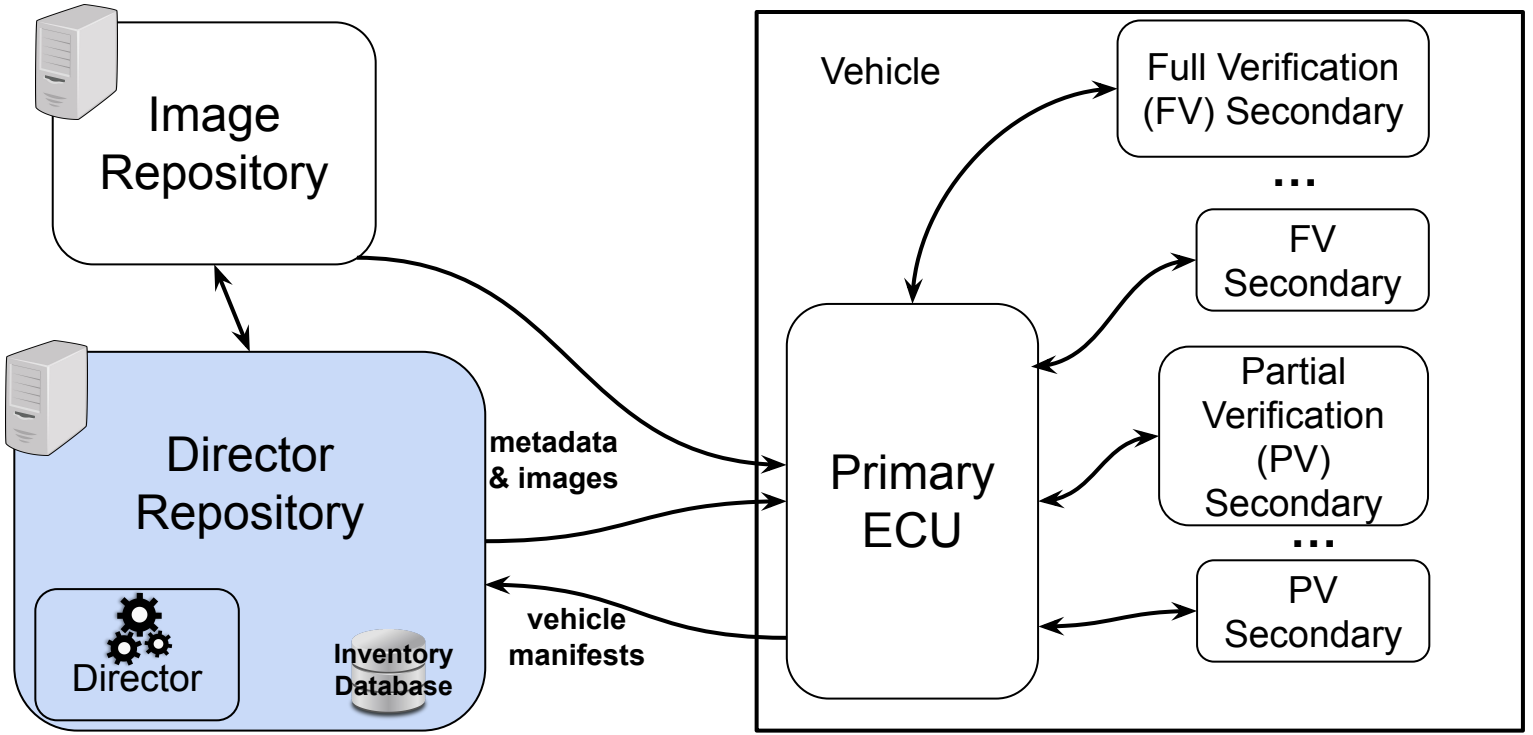
- Designed as a collaboration between researchers and industry experts
- Participation from vehicle manufacturers and suppliers from around the world

https://uptane.github.io/papers/uptane-standard.1.2.0.html

Uptane design - the vehicle



Uptane design - the ecosystem



Uptane community organization

- Open to everyone
  - Open source
  - Patent free
  - Standardized but not prescriptive
- Security of SOTA operations, not SOTA technology

Uptane standardization

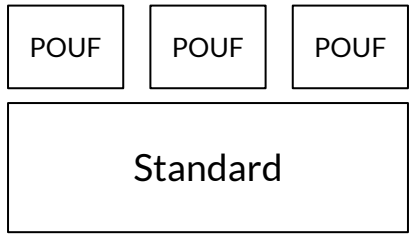
- First Uptane UMTRI/NYU workshop in February 2016 - three years of DHS funding.
- First Uptane Standard 1.0.0 released in July 2019 as IEEE/ISTO 6100.1.0.0.
- Uptane project moved to Linux Foundation's Joint Development Fund in Fall 2019.
- First Uptane Standard 1.1.0 editorial update released in January 2021.
- Second Uptane Standard 1.2.0 editorial update released in July 2021.
- Companion Uptane Deployment Best Practices volumes w/ each standard release.

Uptane Standard for Design and Implementation v1.2.0

uptane-standard-design

Uptane POUFs (Protocols, Operations, Usage, and Formats)

- A profile layer on top of the Uptane Standard
- Allows for interoperable Uptane implementations
- Describes an implementation
  - Choices made from the Uptane Standard and Deployment Considerations
  - Networking information, file storage and data definitions



POUF

Section	Purpose
Abstract	Overview of the POUF
Protocols	Networking information, data transmission, and data binding format
Operations	Design decisions and added features
Usage	Implementation management aspects, including key management, server setup, and data storage strategies
Formats	Data definitions and ordering

Uptane integrations

- Many top suppliers / vendors are adopting Uptane for future cars!
    - Millions of cars on roads worldwide
- Automotive Grade Linux
- OEM integrations
    - Easy to integrate!
    - Migrations from legacy update systems

AUTOMOTIVE GRADE LINUX

FOUNDRIES.IO

Airbiquity OTAmatic®

Uptane security audits

Reviews of implementations and design:

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

Uptane is a living standard

Future areas include:

- Software supply chain security
- Guidance for aftermarket updates and right-to-repair
- Government emergency updates

Teaser for the next talk

- Options for partial verification secondaries
- Migrations from legacy systems
- Stakeholder feedback
- Open challenges

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/

Questions

Marina Moore
marinamoore@nyu.edu

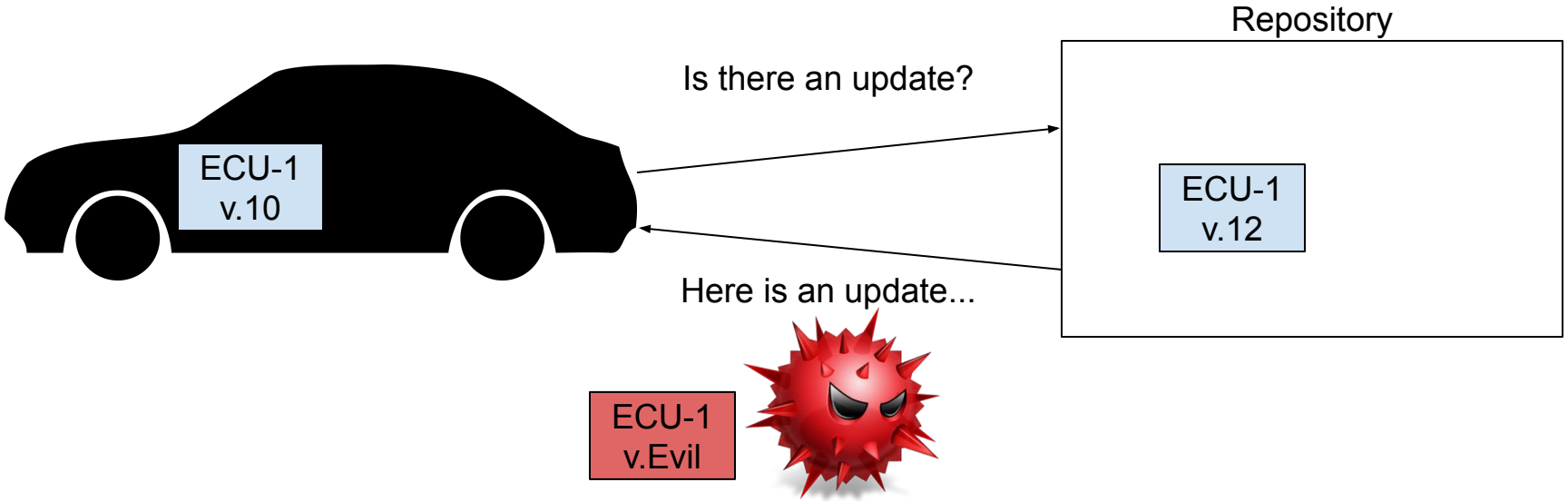
Ira McDonald
blueroofmusic@gmail.com

Uptane.github.io

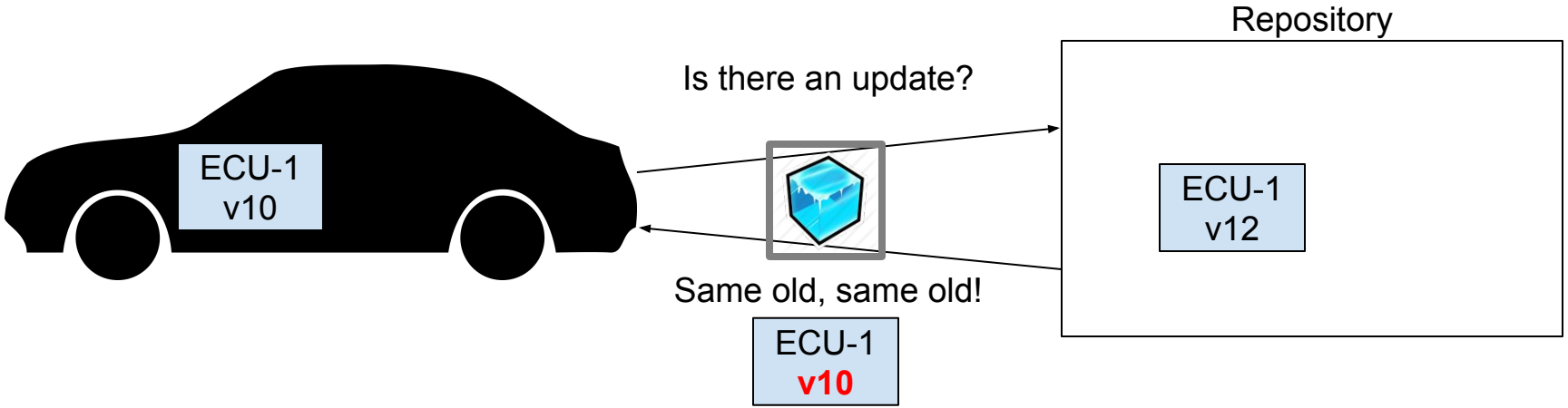
https://uptane.github.io/uptane-standard/uptane-standard.html

Backup slides

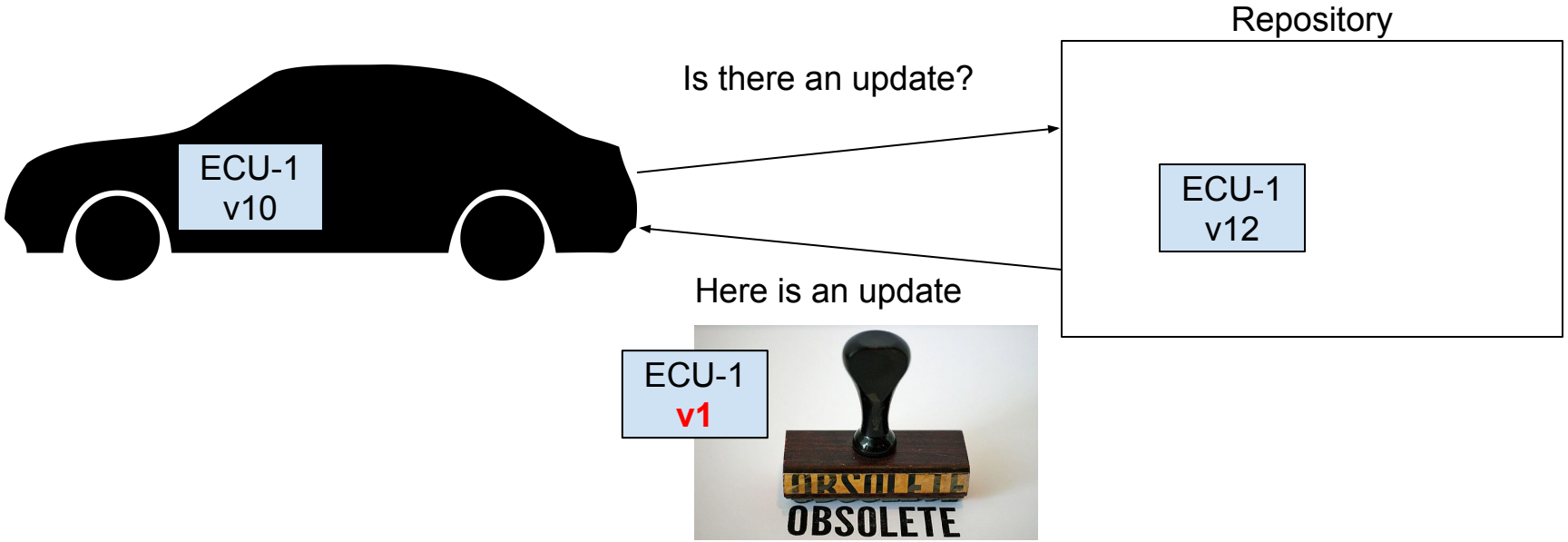
Arbitrary software attack



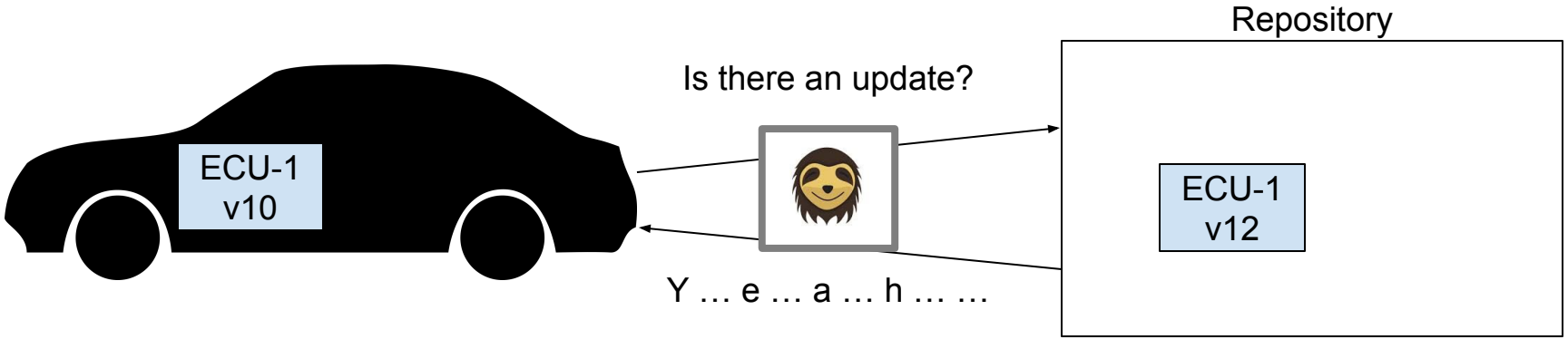
Freeze attack



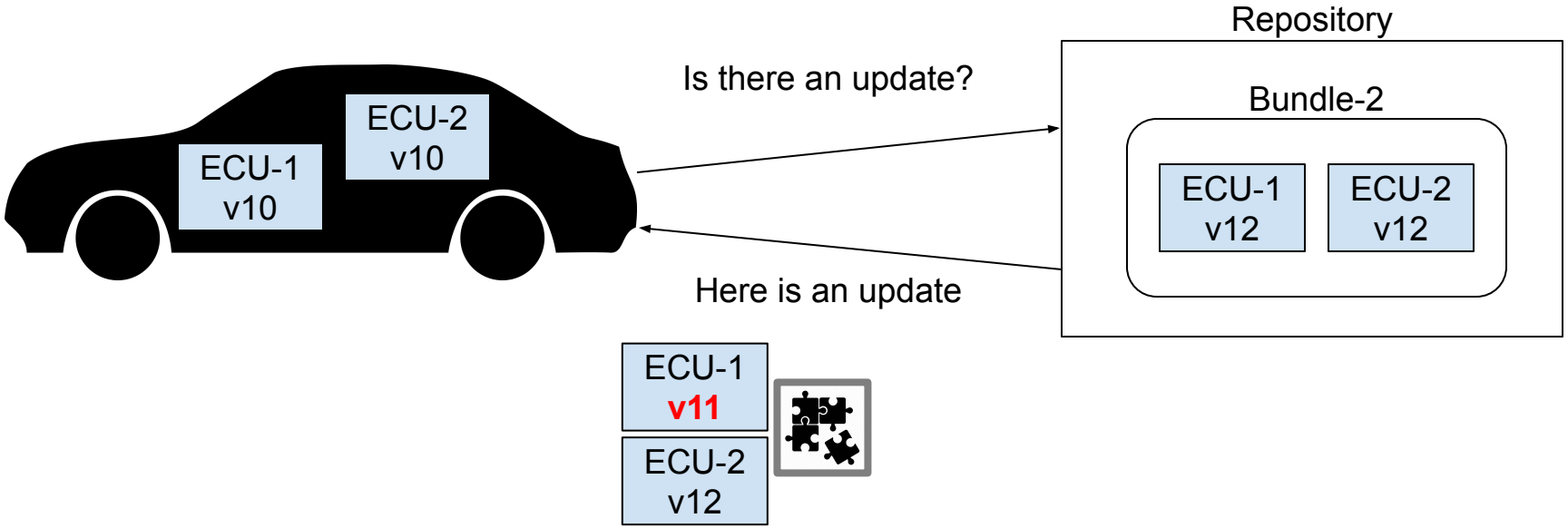
Rollback attack



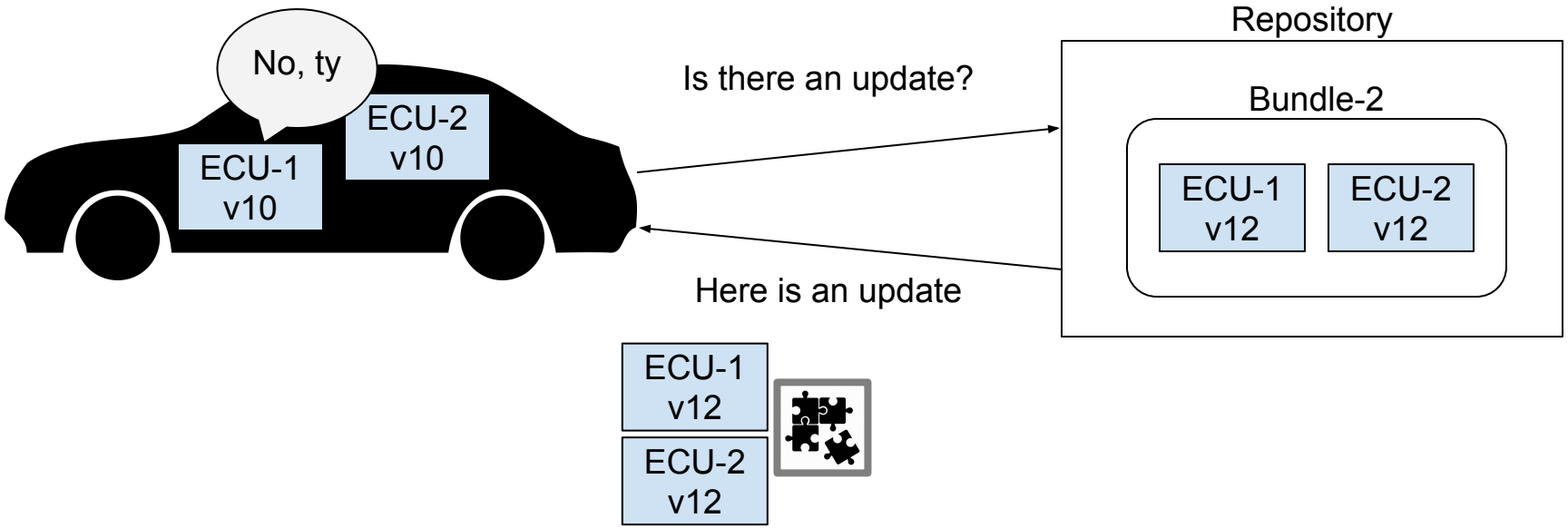
Slow retrieval attack



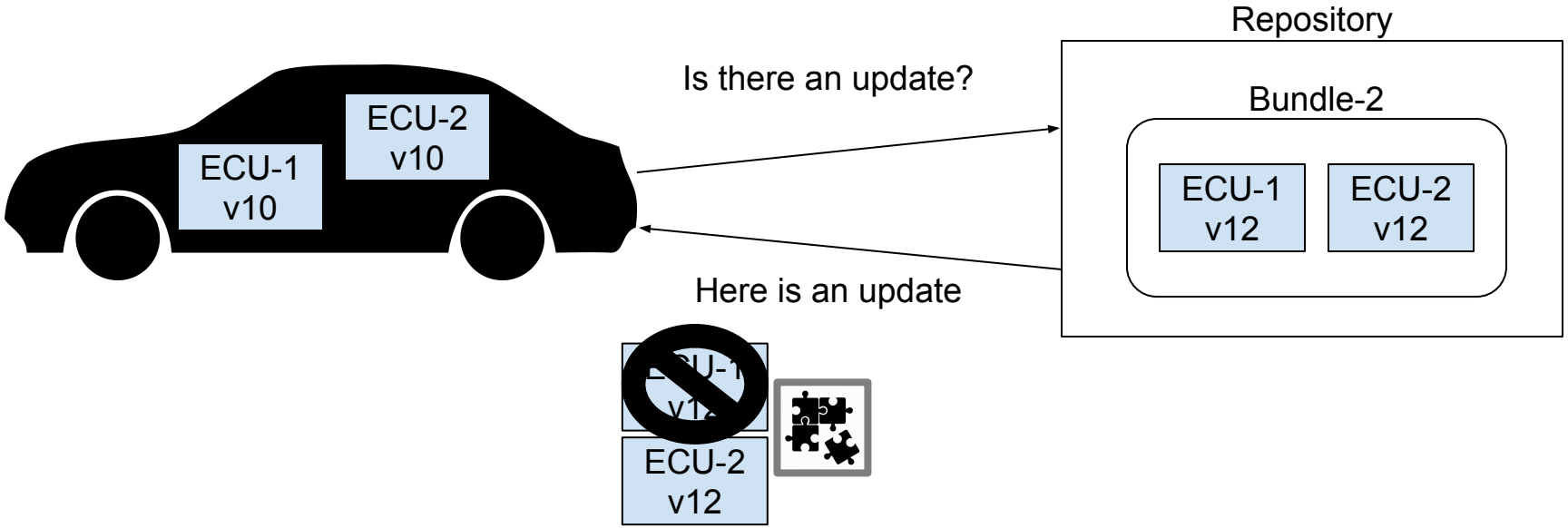
Mix and Match attacks



Partial Bundle attack



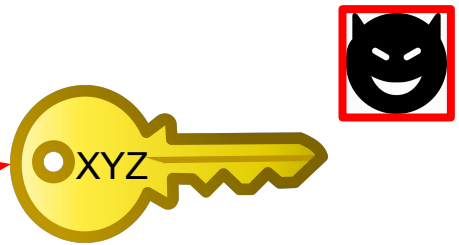
Partial Freeze attack



Inadequate Update Security 4: Just Sign!

Traditional Solution 2:

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

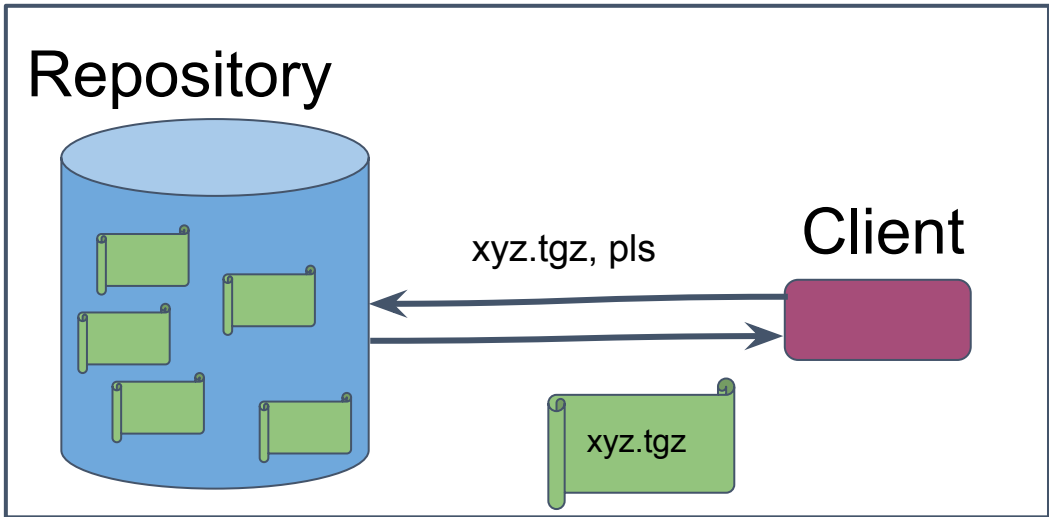


Client has to trust this key

… 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.

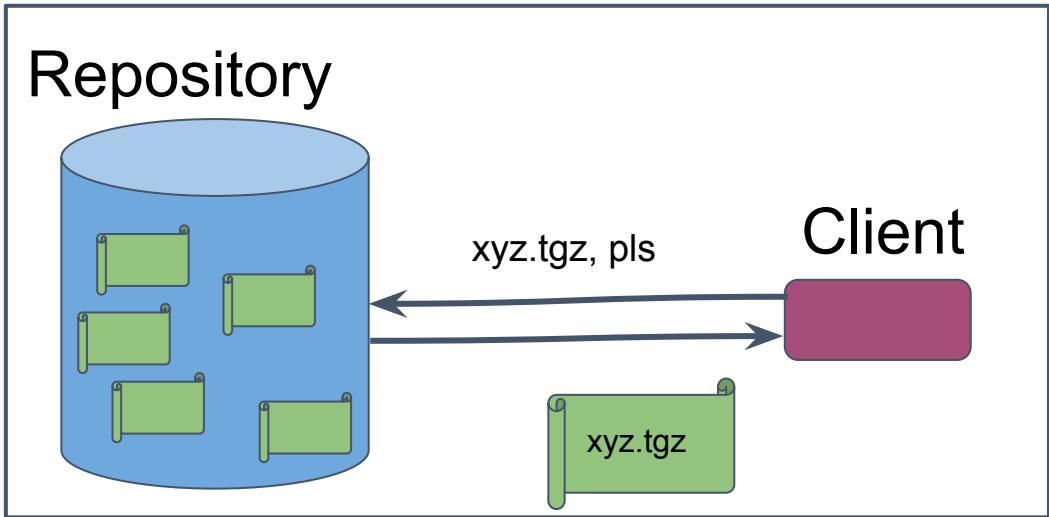


Update Security

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 known attacks on software update systems

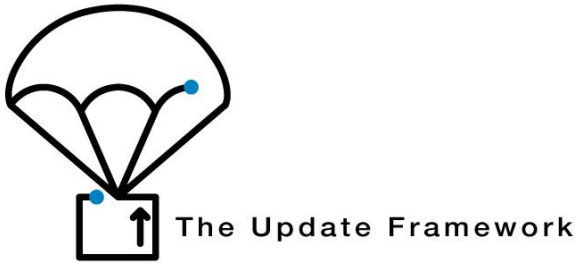
Must not have single point of failure!



The Update Framework (TUF): Goals

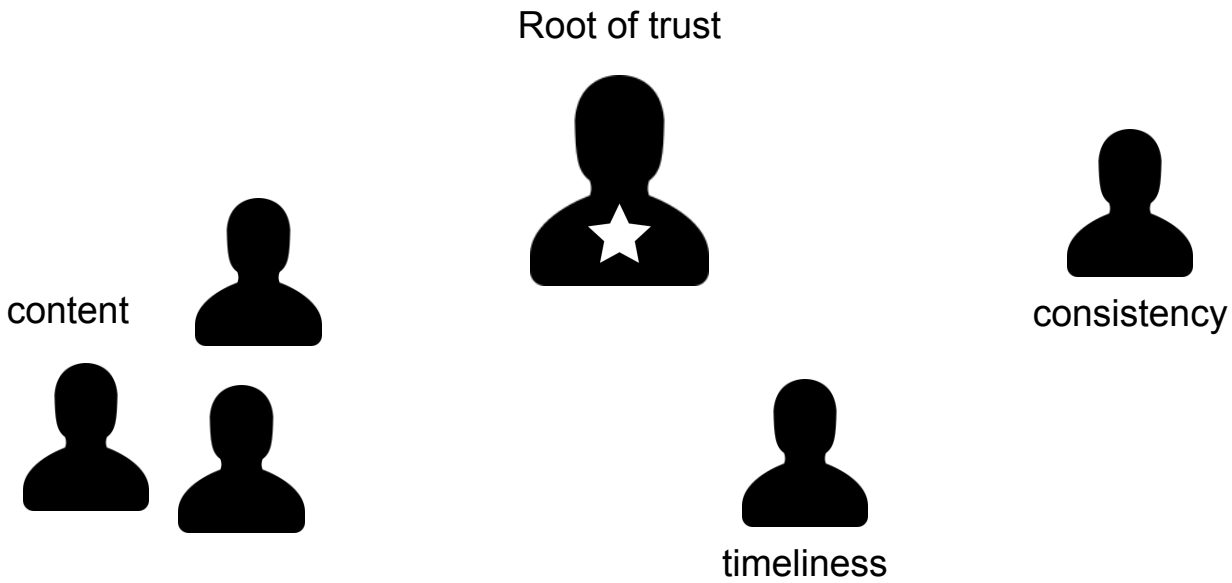
TUF goal “Compromise Resilience”

- 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



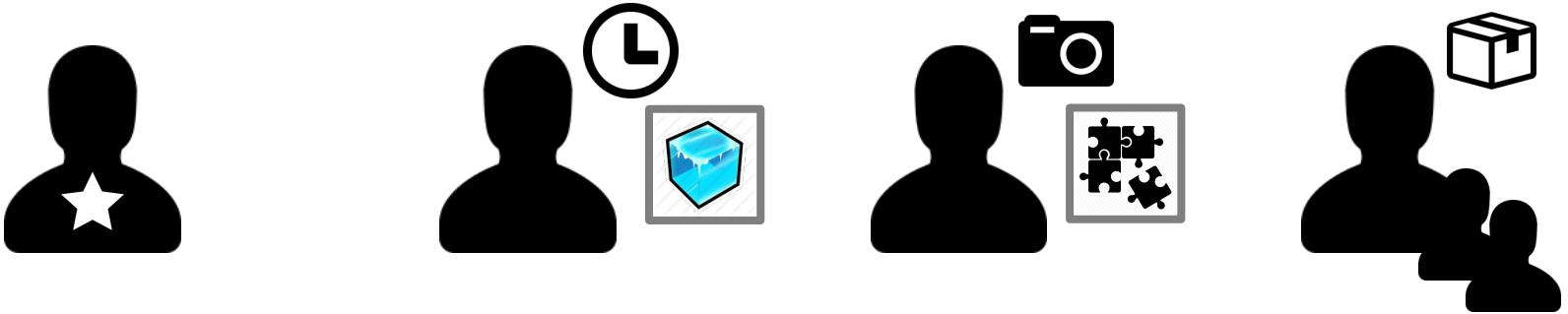
The Update Framework (TUF)

Responsibility Separation



The Update Framework (TUF)

TUF Roles Overview



Root

(root of trust)

Timestamps

(timeliness)

Snapshot

(consistency)

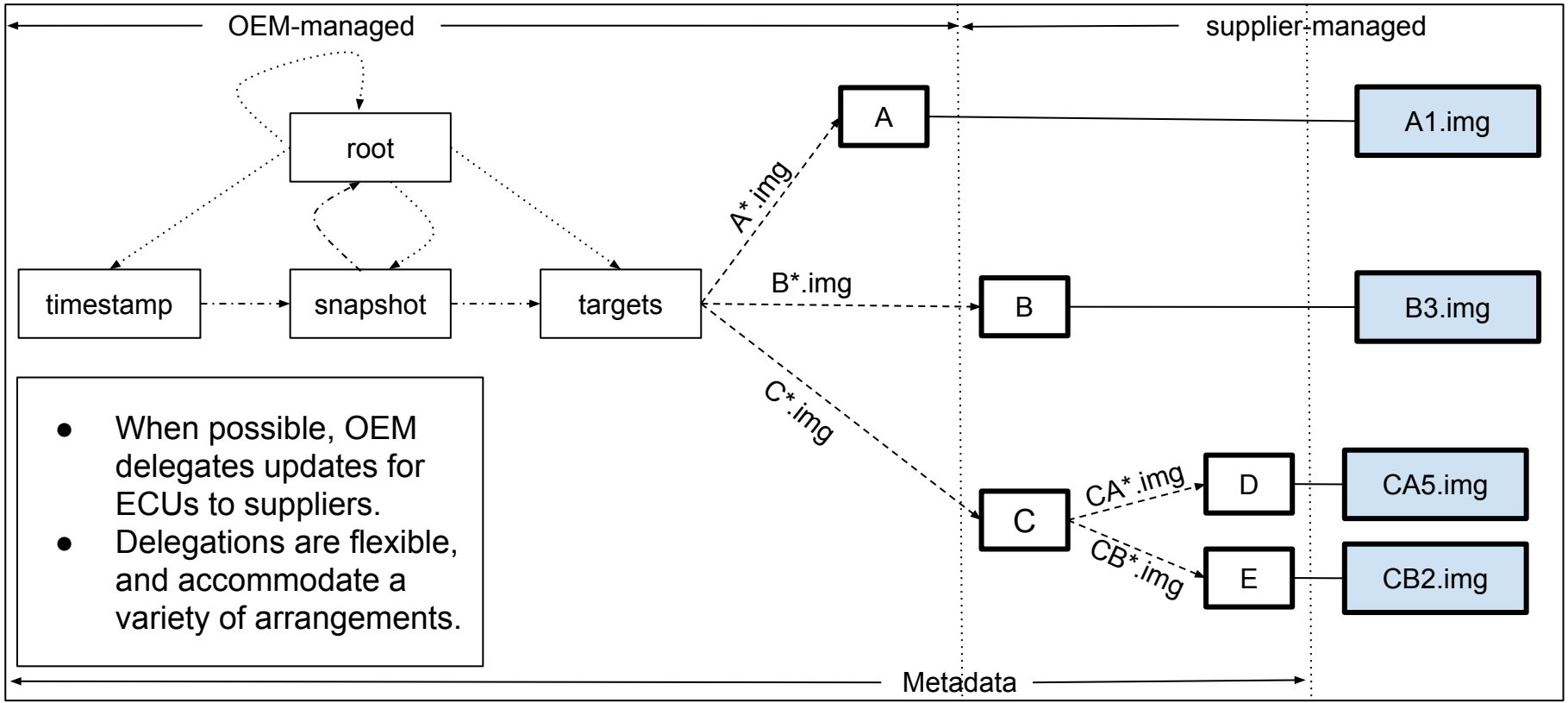
Targets

(integrity)

Uptane builds on The Update Framework (TUF)

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

The image repository



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

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.