MENDER.io

Deploy Software Updates for Linux Devices

Integrate IoT cloud analytics and over-the-air (OTA) updates with Google and Mender.io

Mirza Krak
Embedded Solutions Architect
Mender.io

Session overview

- Over-the-air software updates for IoT and Mender introduction
- Yocto Project introduction
- Google IoT Core and Cloud IoT introduction
- Device authentication integration between Cloud IoT and Mender

About me

- Mirza Krak
  - 8 years in Embedded Linux
    - U-boot and Linux kernel
    - Yocto/Buildroot
  - mirza.krak@northern.tech

- mender.io
  - Open-source update manager for embedded devices
  - Open source (Apache License, v2)
  - Supports a variation of update styles
    - Dual A/B rootfs layout
    - Update Modules (beta)
  - Remote deployment management (server)
  - Under active development

We are hiring

Zener

CFEngine™

https://northern.tech/careers

Northern.tech

MENDER

Internet of Things (IoT)

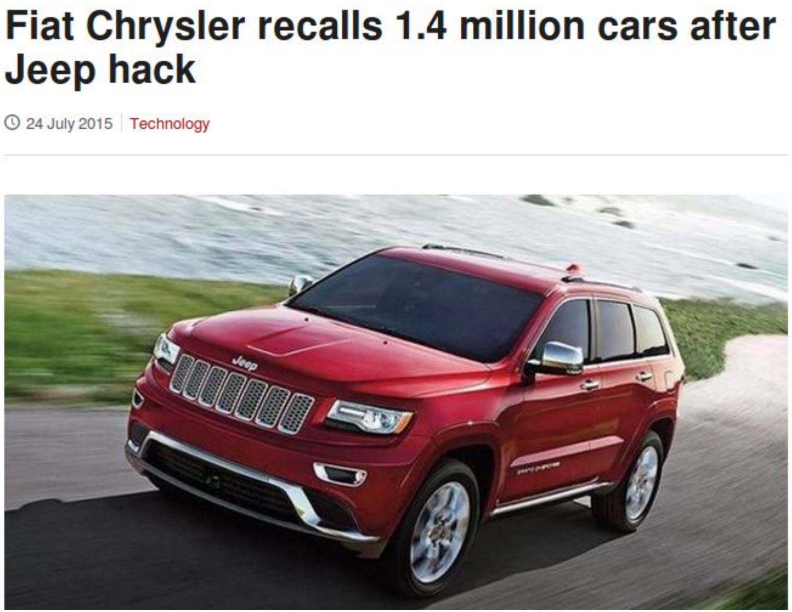
The Internet of things (IoT) is the extension of Internet connectivity into physical devices and everyday objects. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with others over the Internet, and they can be remotely monitored and controlled

Source: Wikipedia

It means taking all the things in the world and connecting them to the internet

Connected devices must be remotely updatable

- There *will* be **bugs, vulnerabilities**
  - 1-25 per 1000 lines of code*
  - Botnets w/ millions of devices: Mirai, Hajime, Brickerbot
- … and new **features**
- … after device is **deployed to the field**



Fiat Chrysler recalls 1.4 million cars after Jeep hack

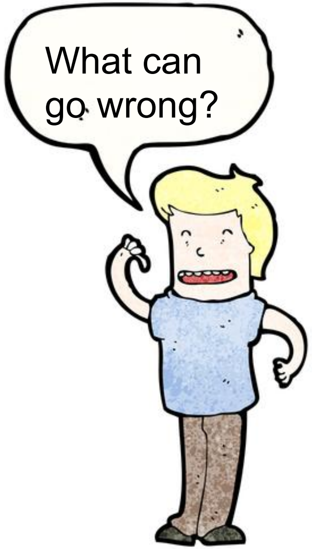
24 July 2015 | Technology

Source: Ars Technica

**Source: Steve McConnell, Code Complete*

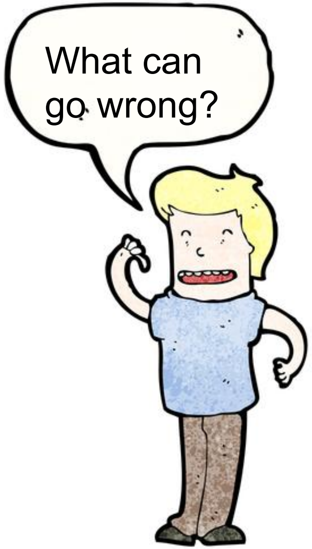
IoT devices are in a harsh environment

- Remote
  - Expensive to reach physically
- Long expected lifetime
  - 5 - 10 years
- Unreliable power
  - Battery
  - Suddenly unplugged
- Unreliable network
  - Intermittent connectivity
  - Low bandwidth
  - Insecure

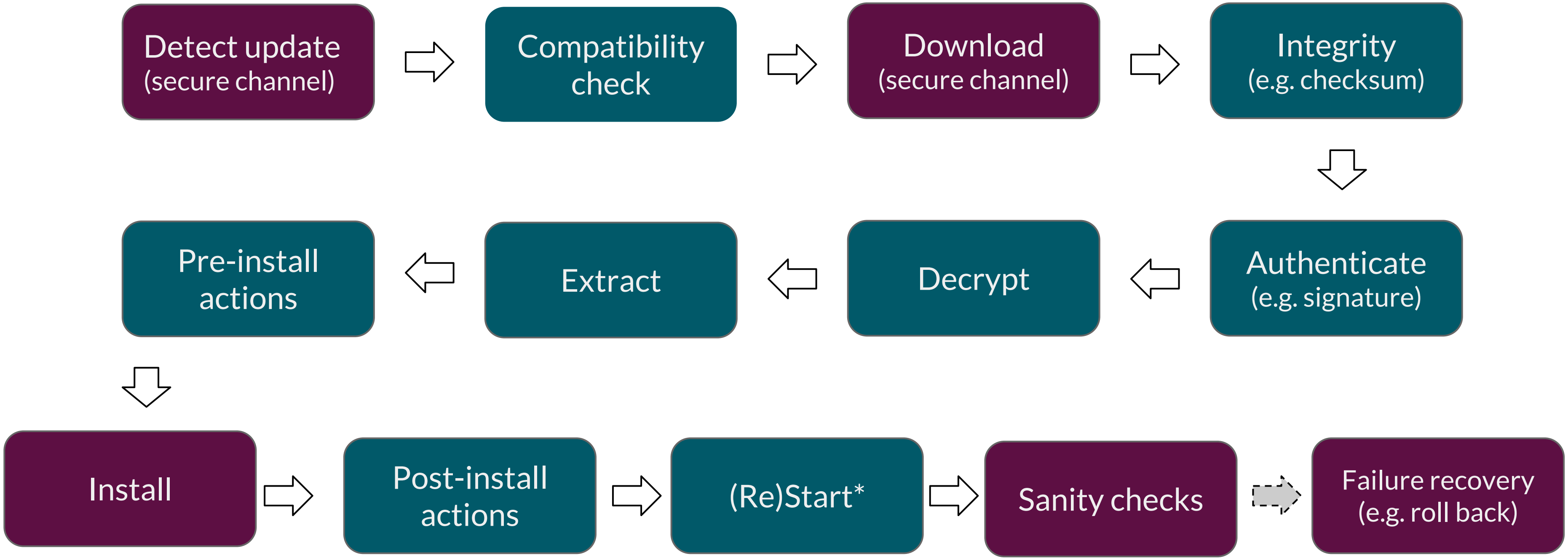


Criteria for IoT software update management

- Robust and secure
- Atomic installation & consistent across devices
- Secure transport and codesigning
- Integrates with existing development environment
- Easy to get started
- Bandwidth consumption
- Downtime during update



General IoT update manager workflow



Detect update (secure channel)

Compatibility check

Download (secure channel)

Integrity (e.g. checksum)

Authenticate (e.g. signature)

Decrypt

Extract

Pre-install actions

Install

Post-install actions

(Re)Start*

Sanity checks

Failure recovery (e.g. roll back)

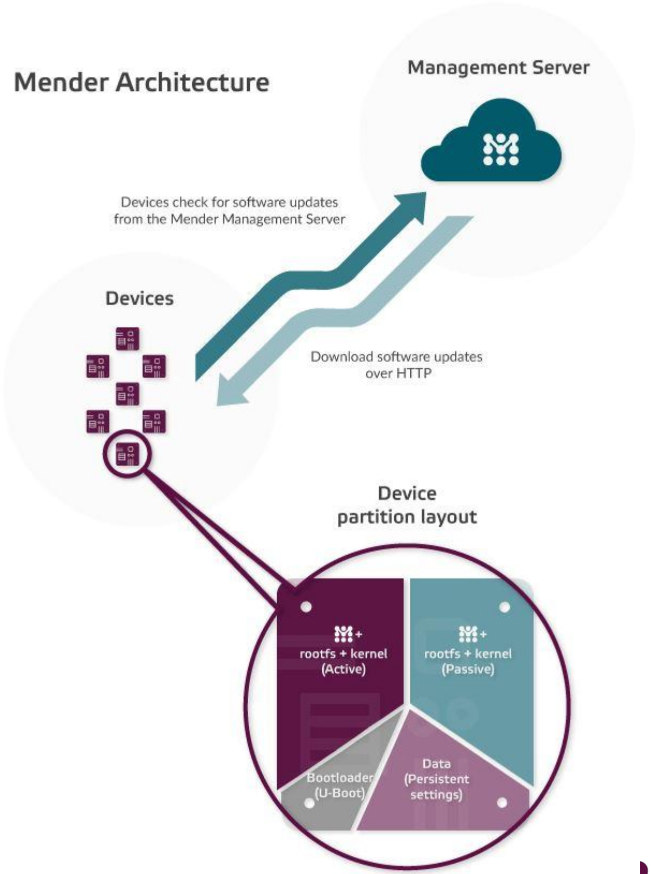
🟪 Must-have

🟦 Environment-specific

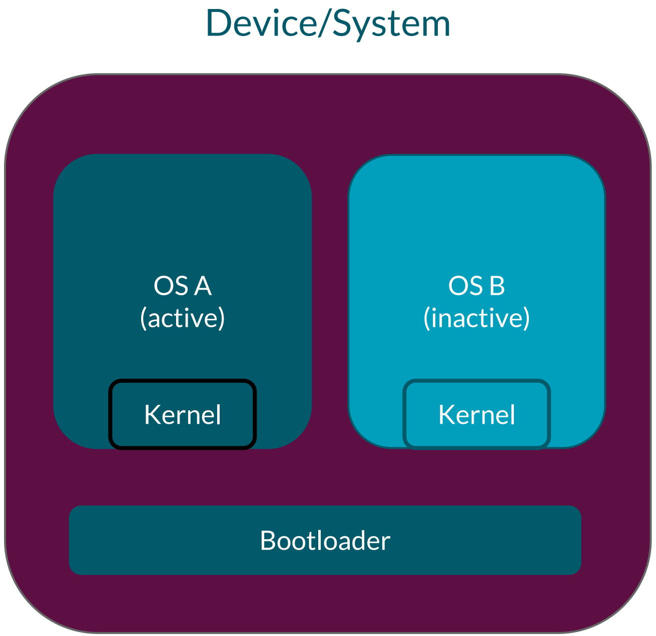
*E.g. reboot, restart service, start container

Mender provides both client and server

- Client-server model
  - Apache 2.0
  - Mender provides both, including web UI
  - No need to “glue” several projects
  - Server can integrate with 3rd party clients through its REST API
- Supports updating
  - File system images
  - Update Modules (beta)
    - Application updates
    - Containers
    - nd more


Mender Architecture
Management Server
Devices check for software updates from the Mender Management Server
Devices
Download software updates over HTTP
Device partition layout
rootfs + kernel (Active)
rootfs + kernel (Passive)
Bootloader (U-Boot)
Data (Persistent settings)

Mender uses a dual A/B system layout



- Mender deploys to inactive partition, then reboots into it
  - Common design for IoT
  - Used in newer Androids (‘N’ and later)

- Very robust
  - Fully atomic and consistent
- Integrates well
  - OS, kernel, apps unchanged
  - Needs bootloader “flip” support
  - Partition layout, requires 2x rootfs storage
- Fairly short downtime (minute)
  - 1 reboot

Mender - server

- Microservices
- Only port 433 and 9000
- RESTful API
  - Device API
  - Management API

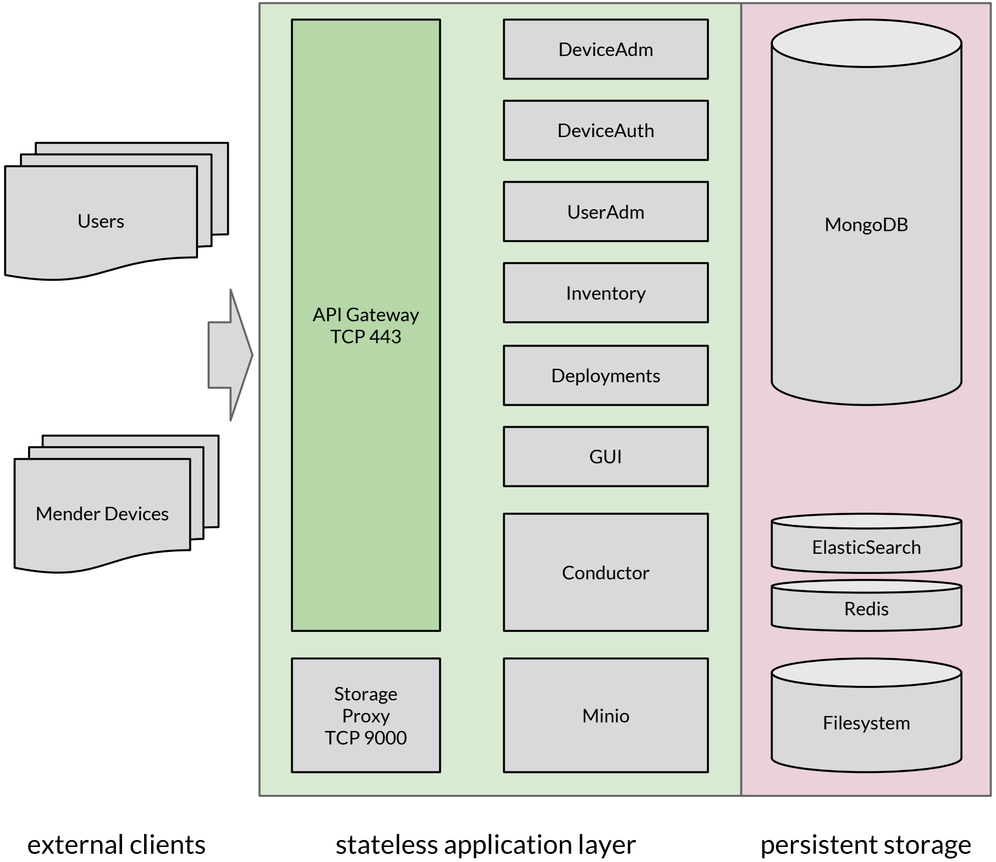
/api/management/v1/deployments

/api/management/v1/admission

/api/management/v1/devauth/

….

https://docs.mender.io/apis/overview



Yocto Project is a Linux build system

“It's not an embedded Linux Distribution, It creates a custom one for you.”

- Structured way to build a Linux distribution from source, using software “meta layers”

- Flexible and very portable between hardware
  - Requires some learning

- Probably the most popular Linux “OS” for IoT devices
  - Major board manufacturers provide BSPs as Yocto meta layers

- Mender provides meta-mender for integrating the Mender client

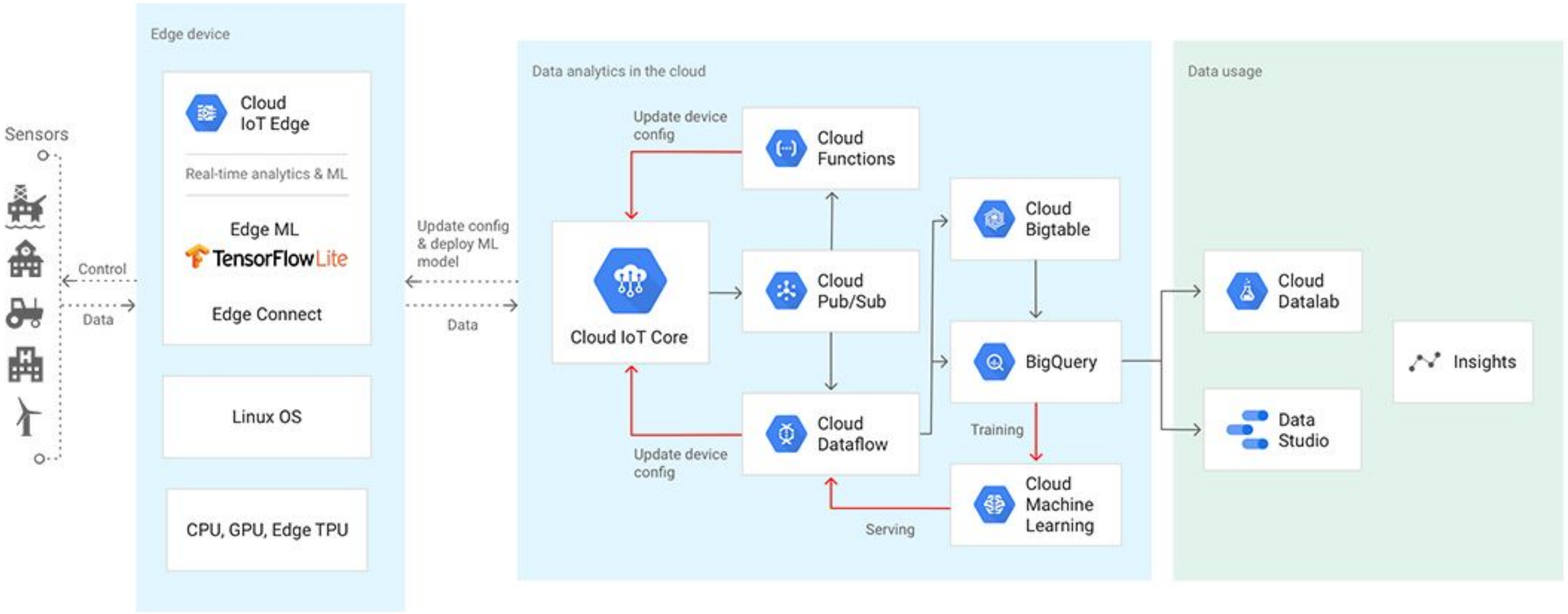
- Google provides meta-gcp-iot for integrating Mender and MQTT telemetry application

Google IoT Core

“Cloud IoT Core is a fully managed service that allows you to easily and securely connect, manage, and ingest data from millions of globally dispersed devices”

- MQTT and HTTP protocols
- scales automatically in response to real-time changes
- industry-standard security protocols protect your data.

Google Cloud IoT (example)



Google IoT Core

Protocol bridge

MQTT protocol endpoint
Automatic load balancing
Global data access with Pub/Sub

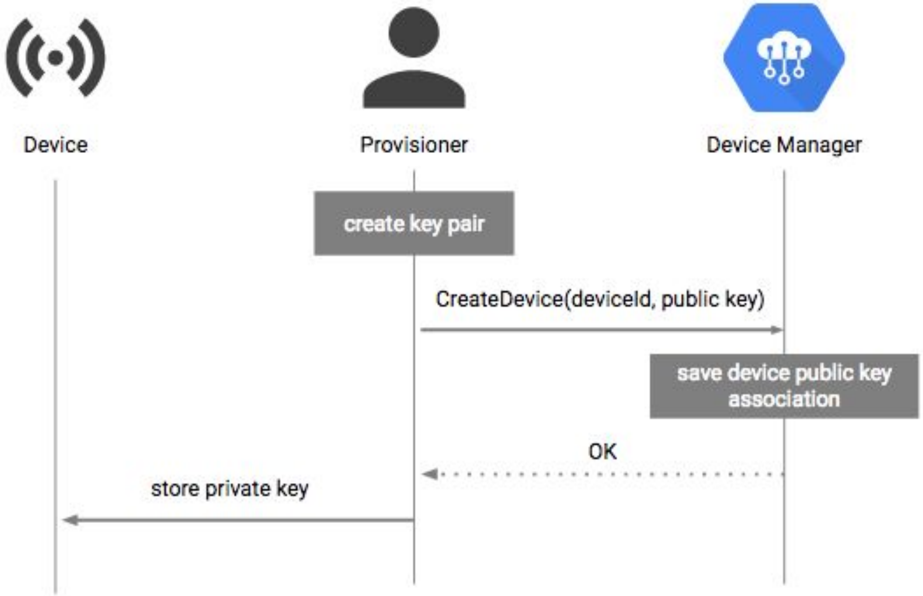
Device manager

Configure individual devices
Update and control devices
Role level access control
Console and APIs for device deployment and monitoring

Device authentication is complex

- To securely authenticate to cloud services, devices need an identity and credential tuple
  - Typically a serial number and public/private keypair
- Different cloud services use different identity and credential tuples
- Result: Identity and key management becomes very complex and error-prone

Device authentication in Google IoT Core


Device
Provisioner
Device Manager
create key pair
CreateDevice(deviceId, public key)
save device public key association
OK
store private key

Device identity is based on an asymmetric key-pair of two supported formats:

- RSA 256 public key wrapped in a X.509v3 certificate
- Elliptic curve (ECDSA) algorithm using P-256 and SHA-256 [more efficient, better suited for small devices]

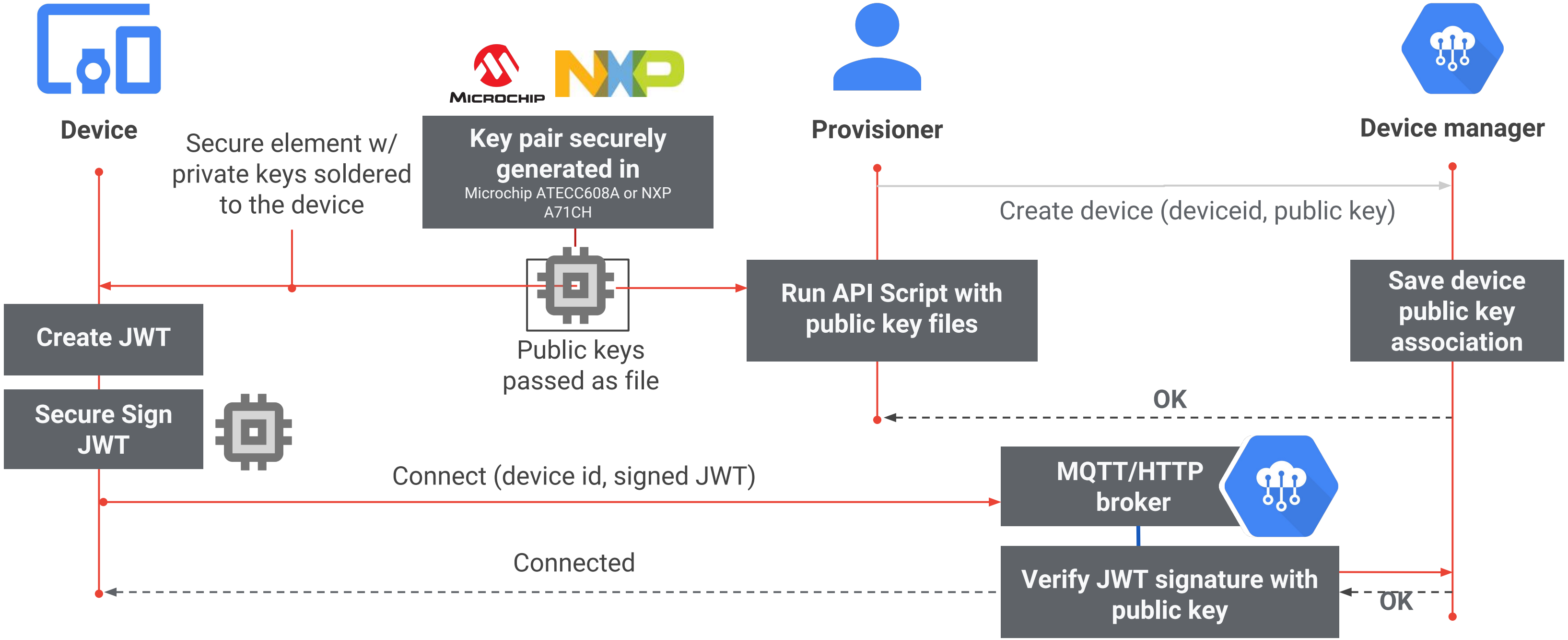
Credentials may optionally have an expiration timestamp

A device can have up to 3 credentials associated with it at a time, allowing for rotation

The service should never need the private key

The sequence shown here is only one way to handle device provisioning

Device authentication in Google IoT Core



Device

Secure element w/ private keys soldered to the device

Key pair securely generated in
Microchip ATECC608A or NXP A71CH

Provisioner

Device manager

Create device (deviceid, public key)

Create JWT

Public keys passed as file

Run API Script with public key files

Save device public key association

Secure Sign JWT

OK

MQTT/HTTP broker

Connect (device id, signed JWT)

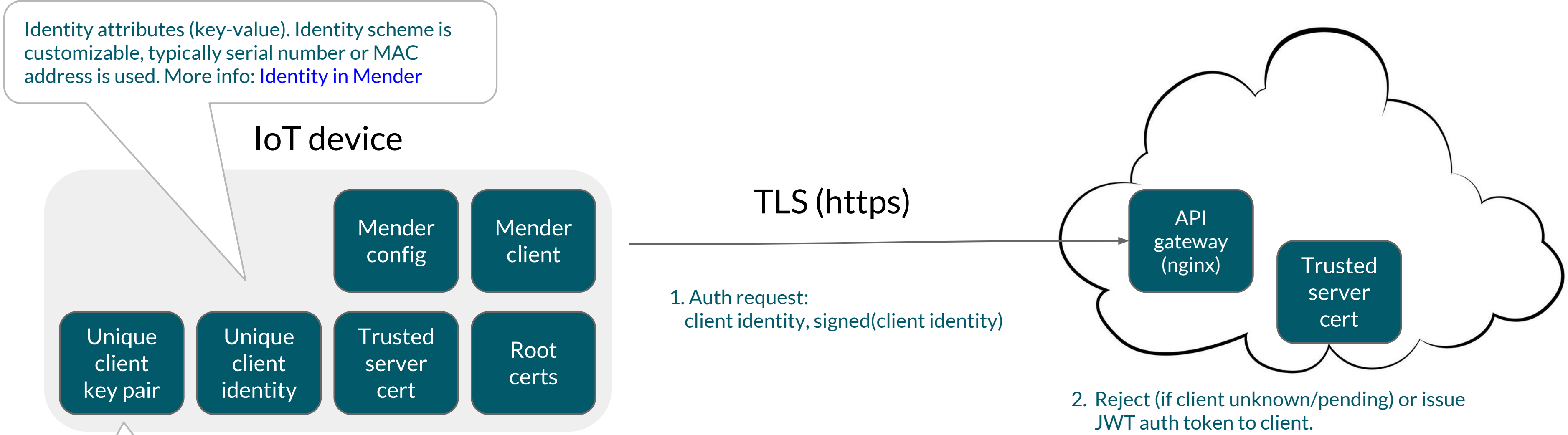
Connected

Verify JWT signature with public key

OK

Device authentication in Mender

Identity attributes (key-value). Identity scheme is customizable, typically serial number or MAC address is used. More info: Identity in Mender



IoT device

Mender config

Mender client

Unique client key pair

Unique client identity

Trusted server cert

Root certs

TLS (https)

1. Auth request:
client identity, signed(client identity)

Mender server

API gateway (nginx)

Trusted server cert

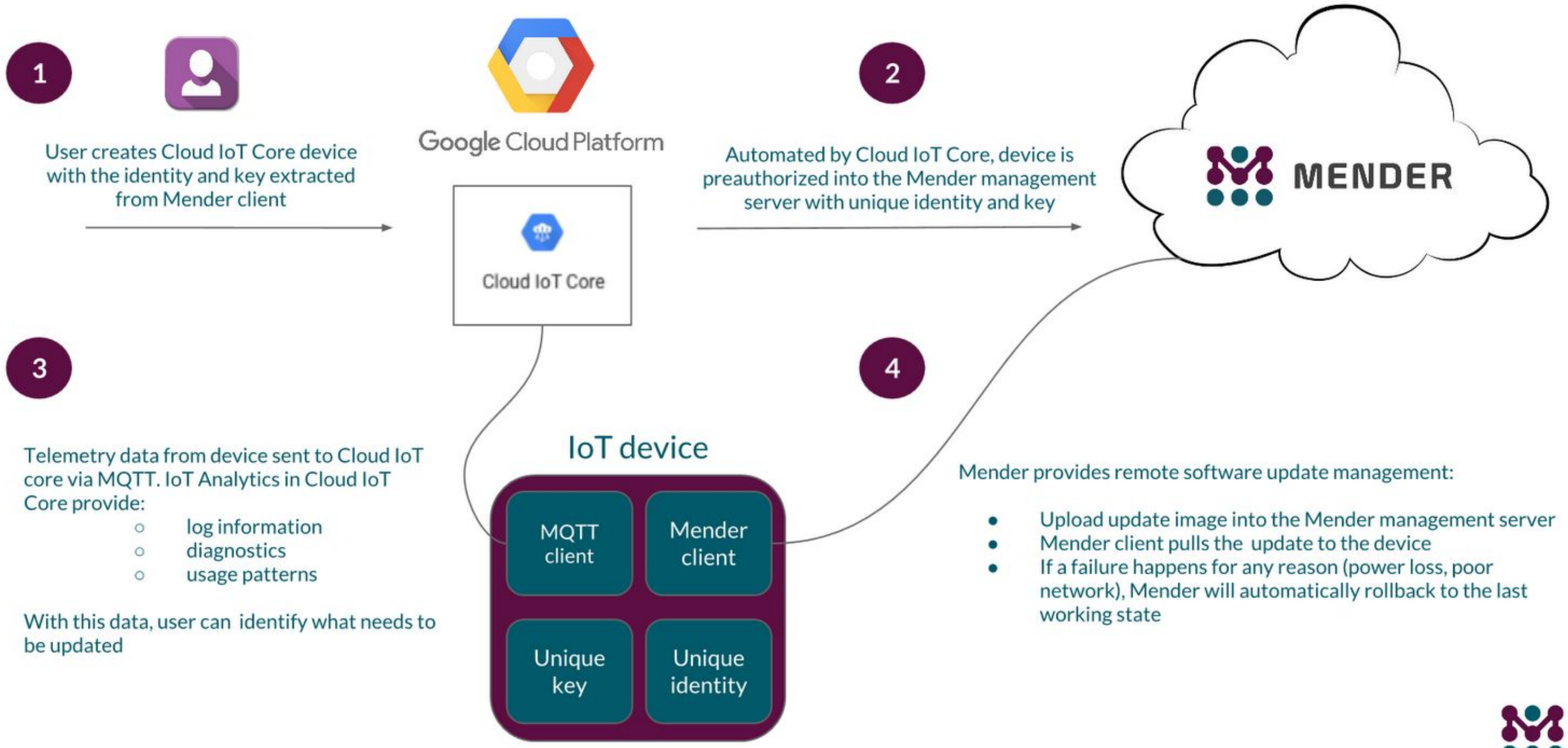
2. Reject (if client unknown/pending) or issue JWT auth token to client.

RSA key unique to this client. Used to sign client identity in auth requests. Will be tied to client identity in server.

Clients get JWT auth token if:

A. They are preauthorized, or
B. Accepted (once pending) by user/script

Device authentication integration workflow



1

User creates Cloud IoT Core device with the identity and key extracted from Mender client

2

Automated by Cloud IoT Core, device is preauthorized into the Mender management server with unique identity and key

3

Telemetry data from device sent to Cloud IoT core via MQTT. IoT Analytics in Cloud IoT Core provide:

- log information
- diagnostics
- usage patterns

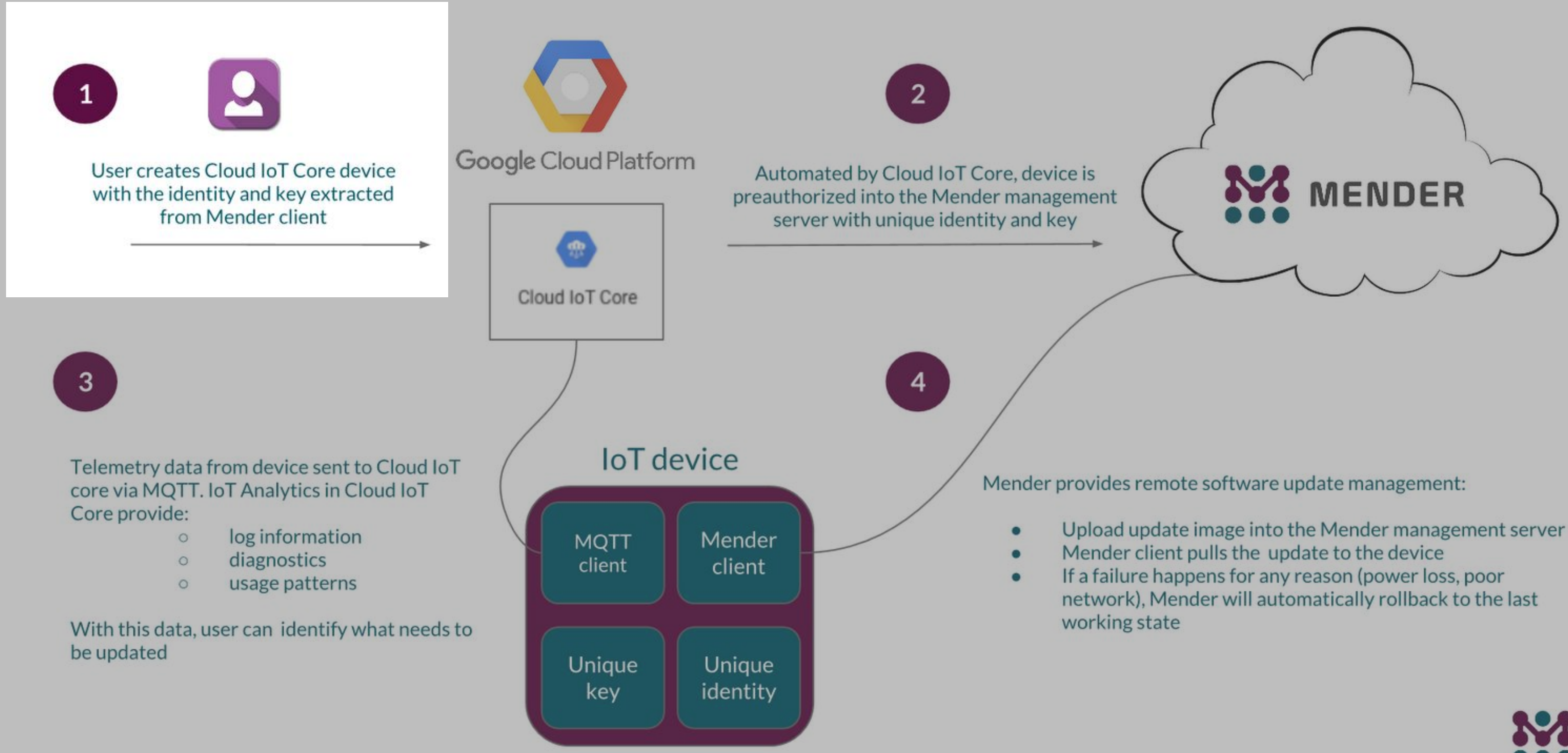
With this data, user can identify what needs to be updated

4

Mender provides remote software update management:

- Upload update image into the Mender management server
- Mender client pulls the update to the device
- If a failure happens for any reason (power loss, poor network), Mender will automatically rollback to the last working state

Device authentication integration workflow



1

User creates Cloud IoT Core device with the identity and key extracted from Mender client

2

Automated by Cloud IoT Core, device is preauthorized into the Mender management server with unique identity and key

3

Telemetry data from device sent to Cloud IoT core via MQTT. IoT Analytics in Cloud IoT Core provide:

- log information
- diagnostics
- usage patterns

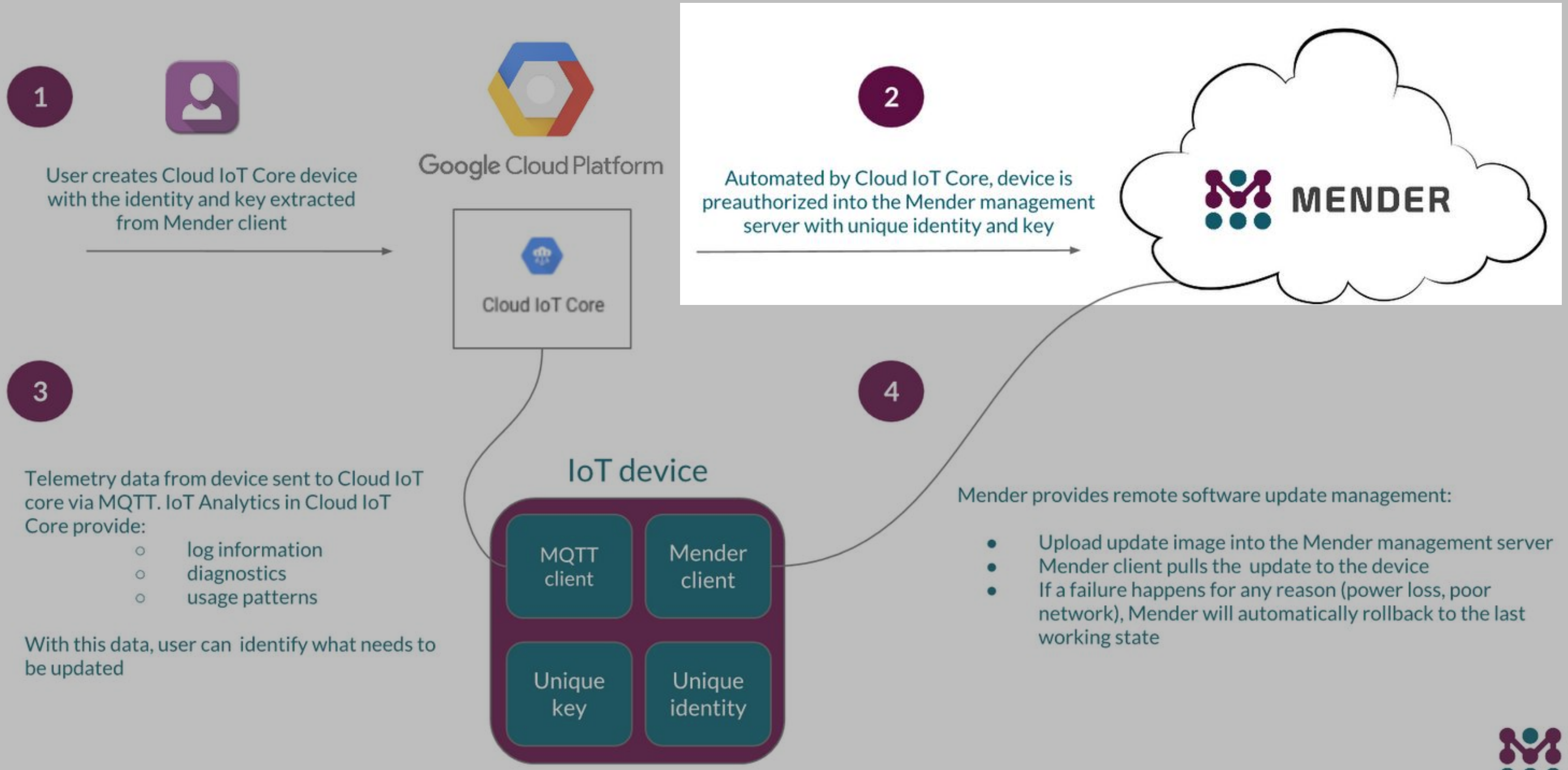
With this data, user can identify what needs to be updated

4

Mender provides remote software update management:

- Upload update image into the Mender management server
- Mender client pulls the update to the device
- If a failure happens for any reason (power loss, poor network), Mender will automatically rollback to the last working state

Device authentication integration workflow



1

User creates Cloud IoT Core device with the identity and key extracted from Mender client

2

Automated by Cloud IoT Core, device is preauthorized into the Mender management server with unique identity and key

3

Telemetry data from device sent to Cloud IoT core via MQTT. IoT Analytics in Cloud IoT Core provide:

- log information
- diagnostics
- usage patterns

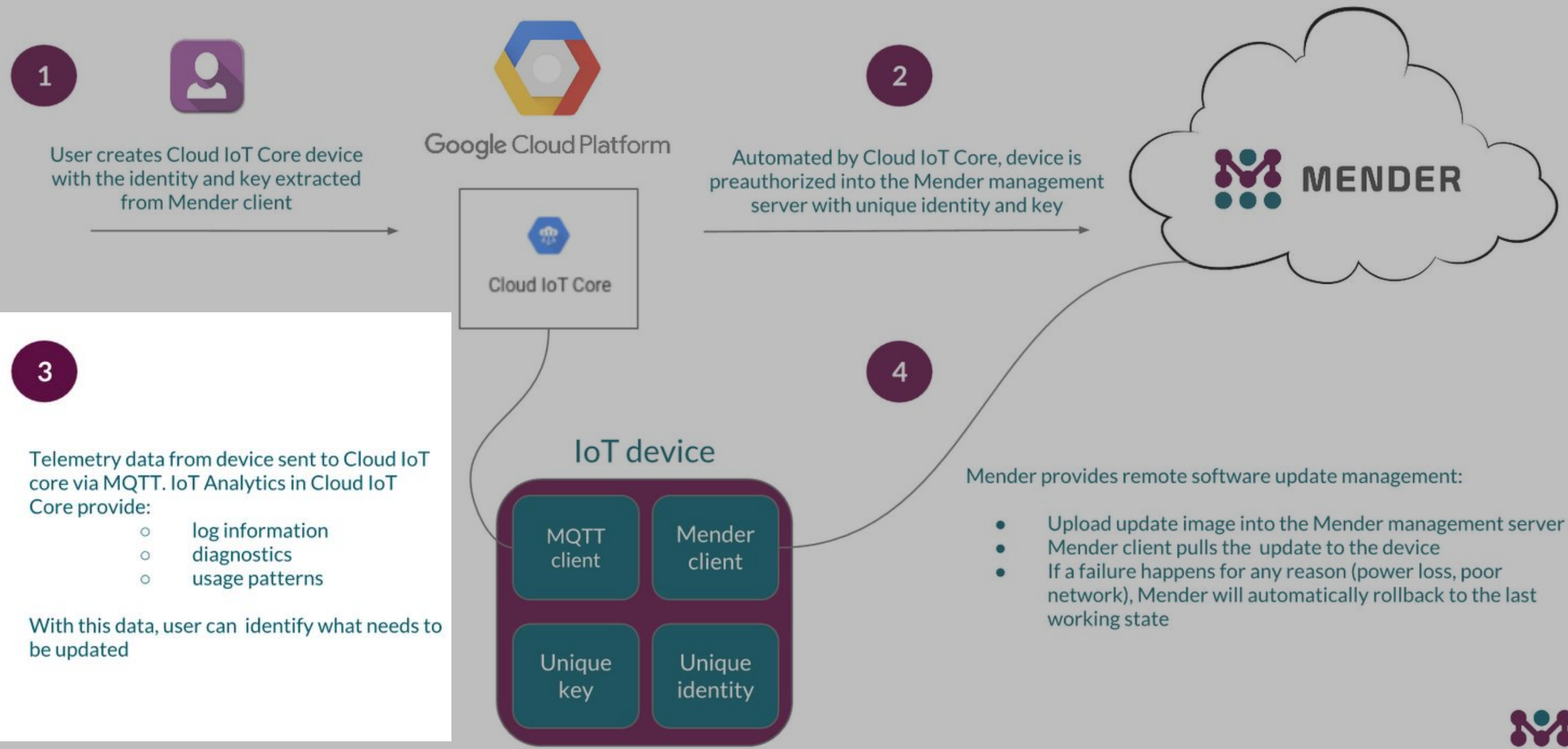
With this data, user can identify what needs to be updated

4

Mender provides remote software update management:

- Upload update image into the Mender management server
- Mender client pulls the update to the device
- If a failure happens for any reason (power loss, poor network), Mender will automatically rollback to the last working state

Device authentication integration workflow



User creates Cloud IoT Core device with the identity and key extracted from Mender client

Automated by Cloud IoT Core, device is preauthorized into the Mender management server with unique identity and key

Telemetry data from device sent to Cloud IoT core via MQTT. IoT Analytics in Cloud IoT Core provide:

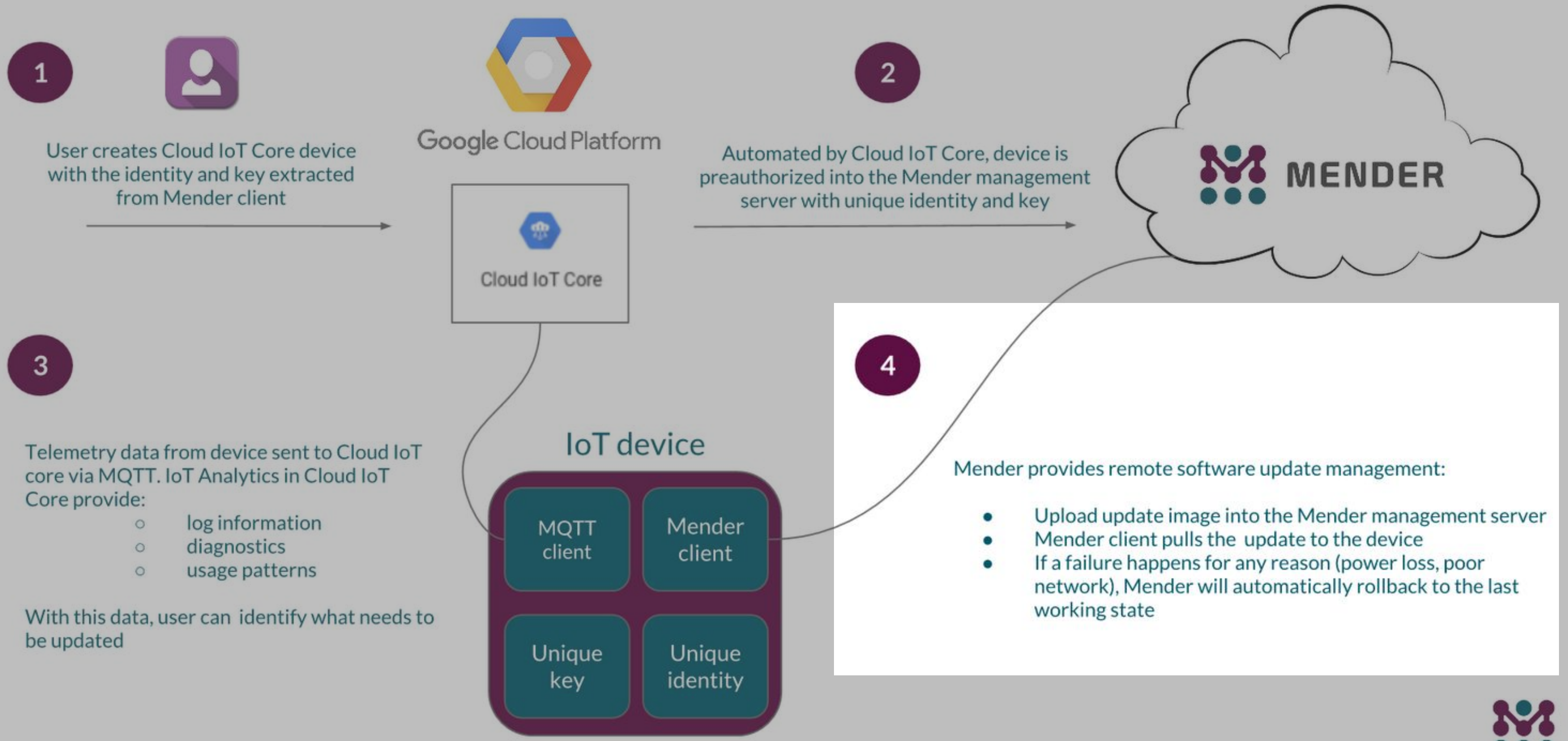
- log information
- diagnostics
- usage patterns

With this data, user can identify what needs to be updated

Mender provides remote software update management:

- Upload update image into the Mender management server
- Mender client pulls the update to the device
- If a failure happens for any reason (power loss, poor network), Mender will automatically rollback to the last working state

Device authentication integration workflow



1

User creates Cloud IoT Core device with the identity and key extracted from Mender client

2

Automated by Cloud IoT Core, device is preauthorized into the Mender management server with unique identity and key

3

Telemetry data from device sent to Cloud IoT core via MQTT. IoT Analytics in Cloud IoT Core provide:

- log information
- diagnostics
- usage patterns

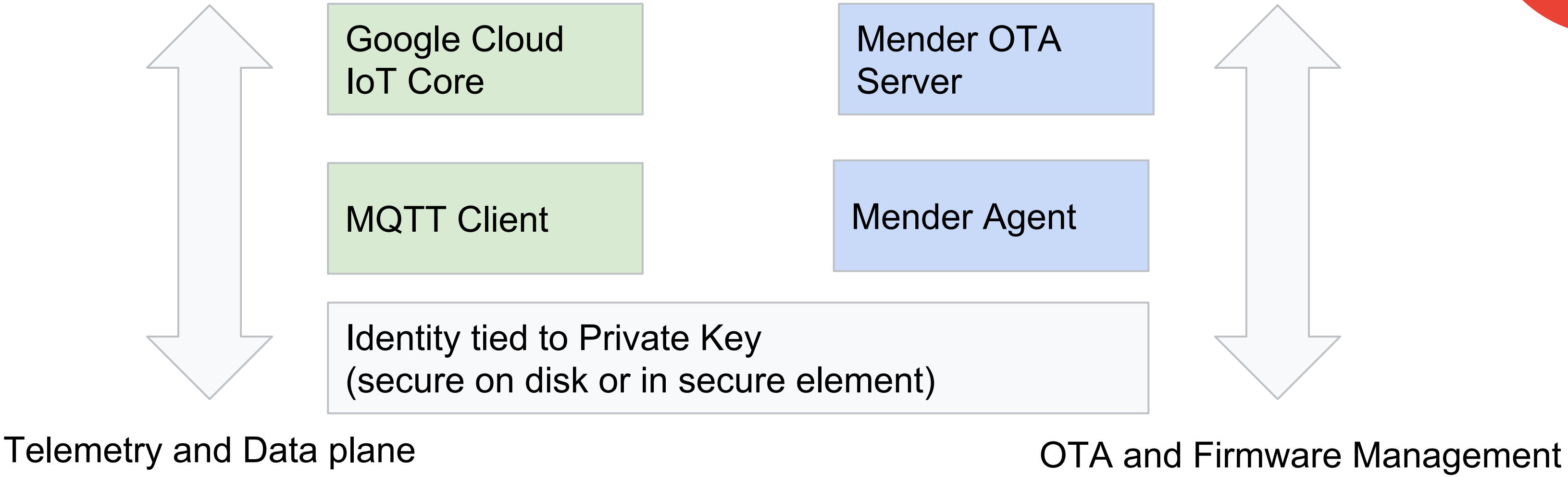
With this data, user can identify what needs to be updated

4

Mender provides remote software update management:

- Upload update image into the Mender management server
- Mender client pulls the update to the device
- If a failure happens for any reason (power loss, poor network), Mender will automatically rollback to the last working state

Integration based on common private key



Google Cloud IoT Core

MQTT Client

Mender OTA Server

Mender Agent

Identity tied to Private Key
(secure on disk or in secure element)

Telemetry and Data plane

OTA and Firmware Management

Reference integration

Step-by-step tutorial available

bit.ly/mender-google

Thank you

Questions?