Dominance as a New Trusted Computing Primitive for the IoT

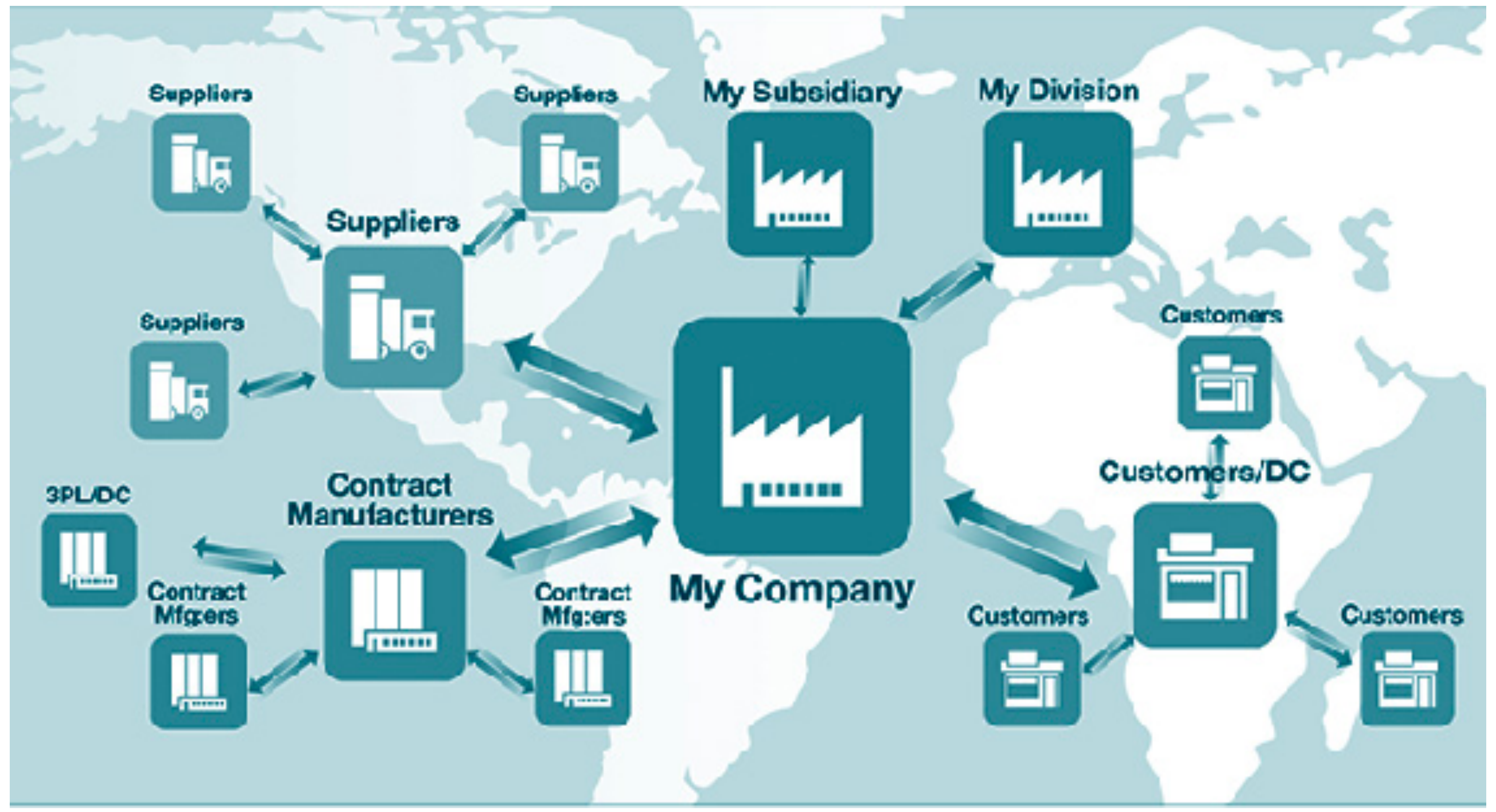
Meng Xu (Georgia Tech)
Manuel Huber (Fraunhofer AISEC)
Zhichuang Sun (Northeastern University)
Paul England (Microsoft Research)
Marcus Peinado (Microsoft Research)
Sangho Lee (Microsoft Research)
Andrey Marochko (Microsoft Research)
Dennis Mattoon (Microsoft Research)
Rob Spiger (Microsoft)
Stefan Thom (Microsoft)

Microsoft Research - Cyber-Resilient Platform Program

Large Scale IoT Deployments Have Arrived

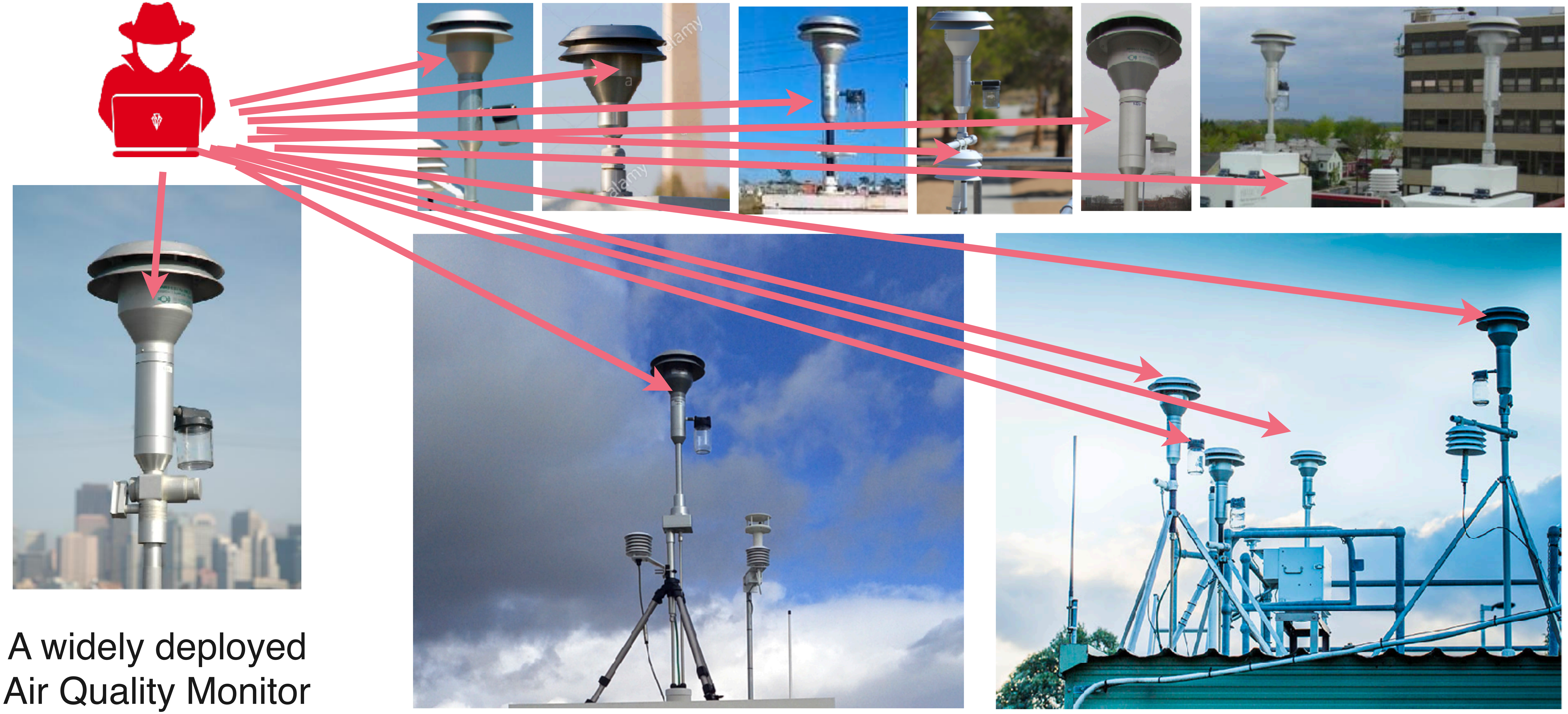
Industrial 4.0

Smart City



Supply Chain

Identical IoT Devices Deployed



A widely deployed Air Quality Monitor

Are We Ready For Large Scale IoT Attacks?

Industrial 4.0

Smart City

Supply Chain

Can we recover a large number of rooted devices without manual intervention?

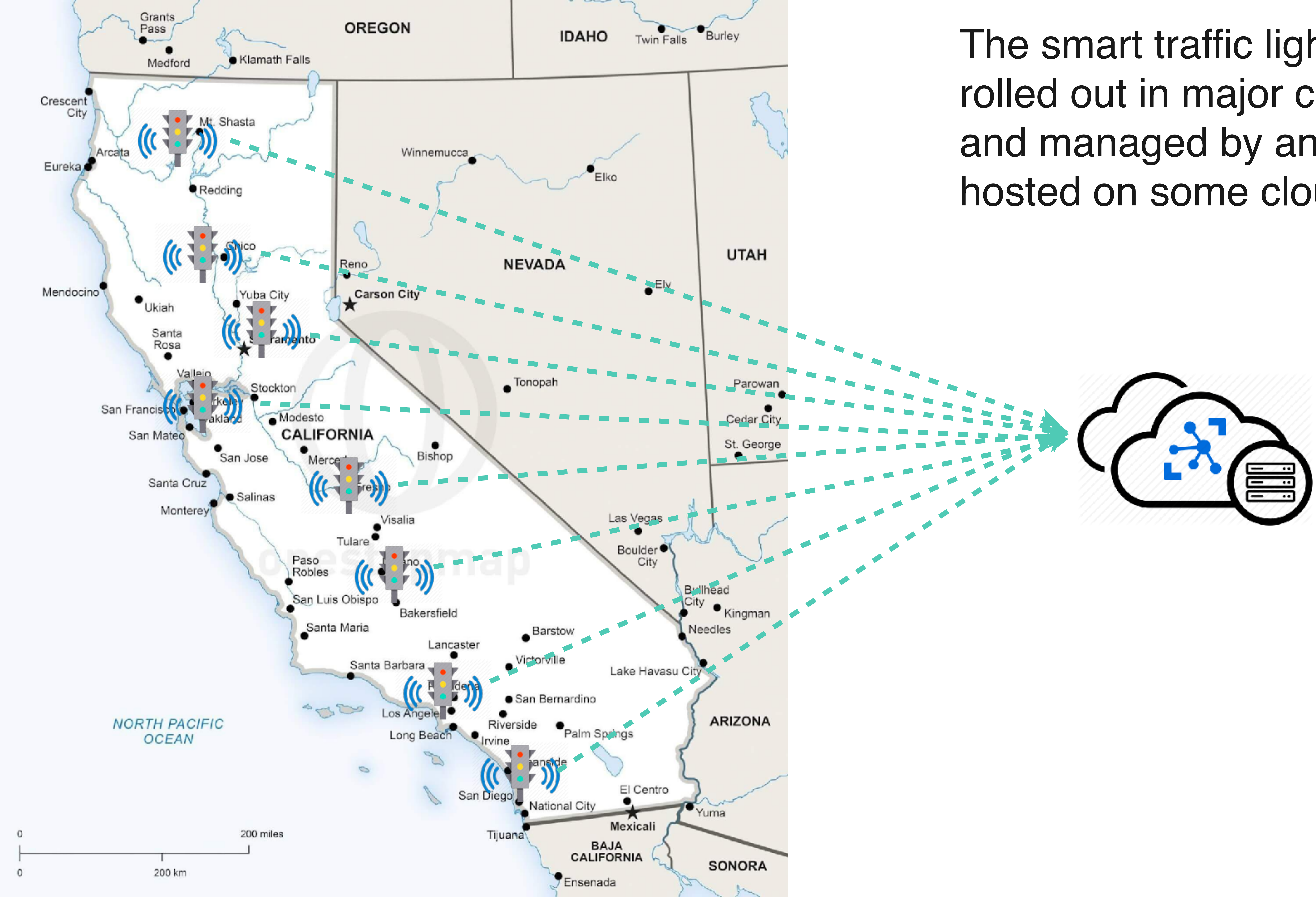
Let’s think this through with a concrete example!

The Tale of California Traffic Lights...

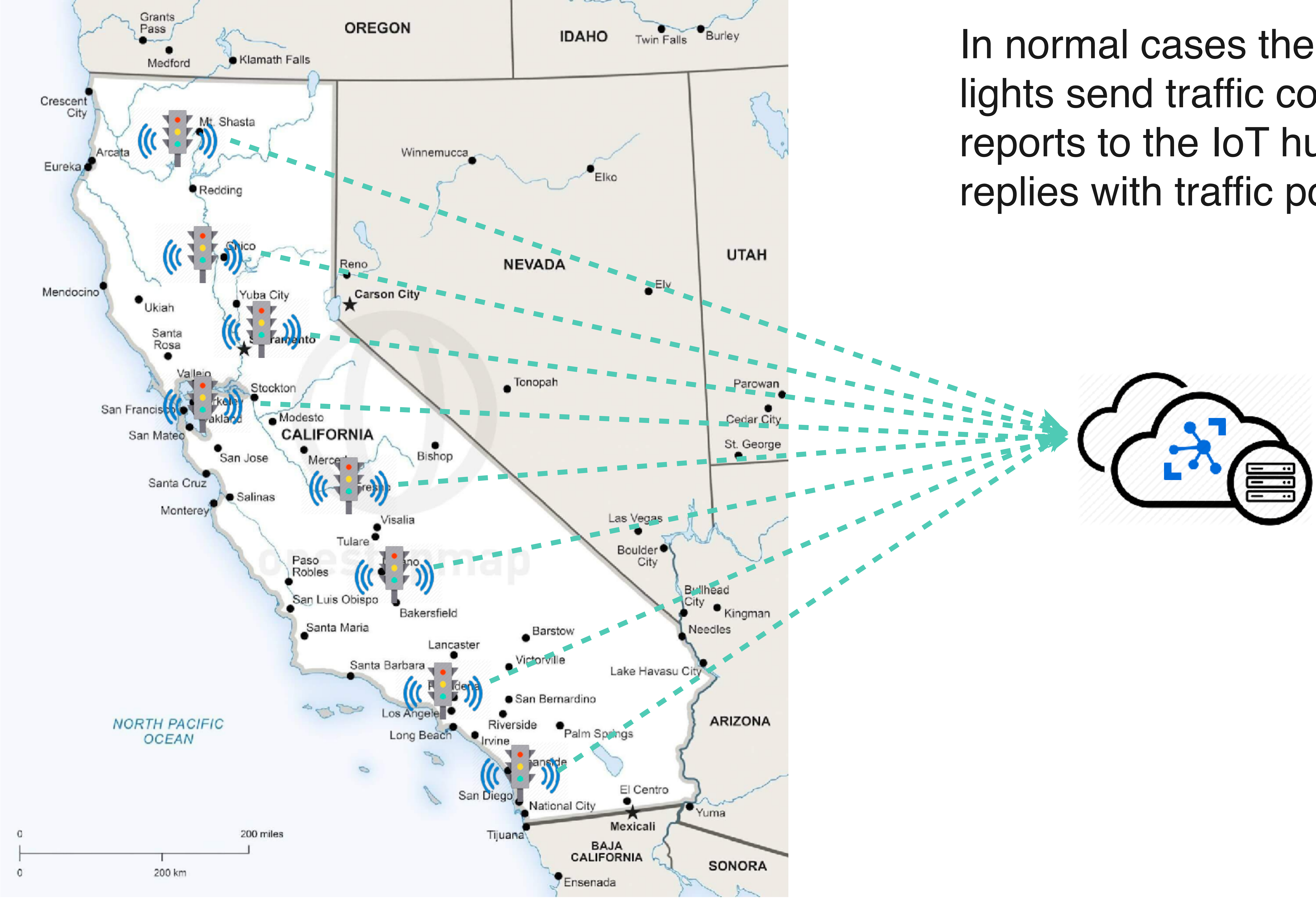
@ Dennis Macdonald—Getty Images

Suppose that your company manages all the smart traffic lights across California.

The smart traffic lights are rolled out in major cities and managed by an IoT hub hosted on some cloud service.

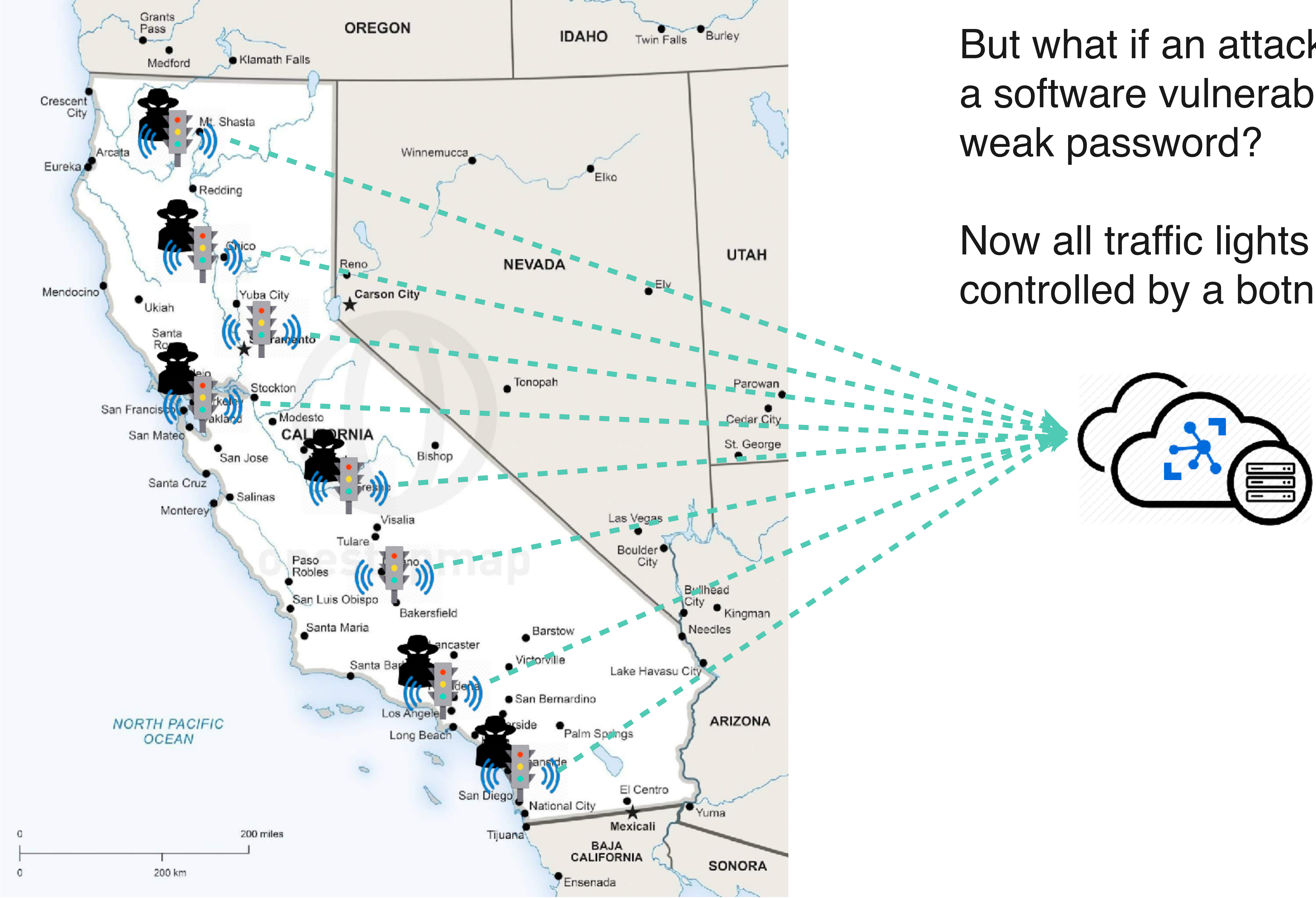


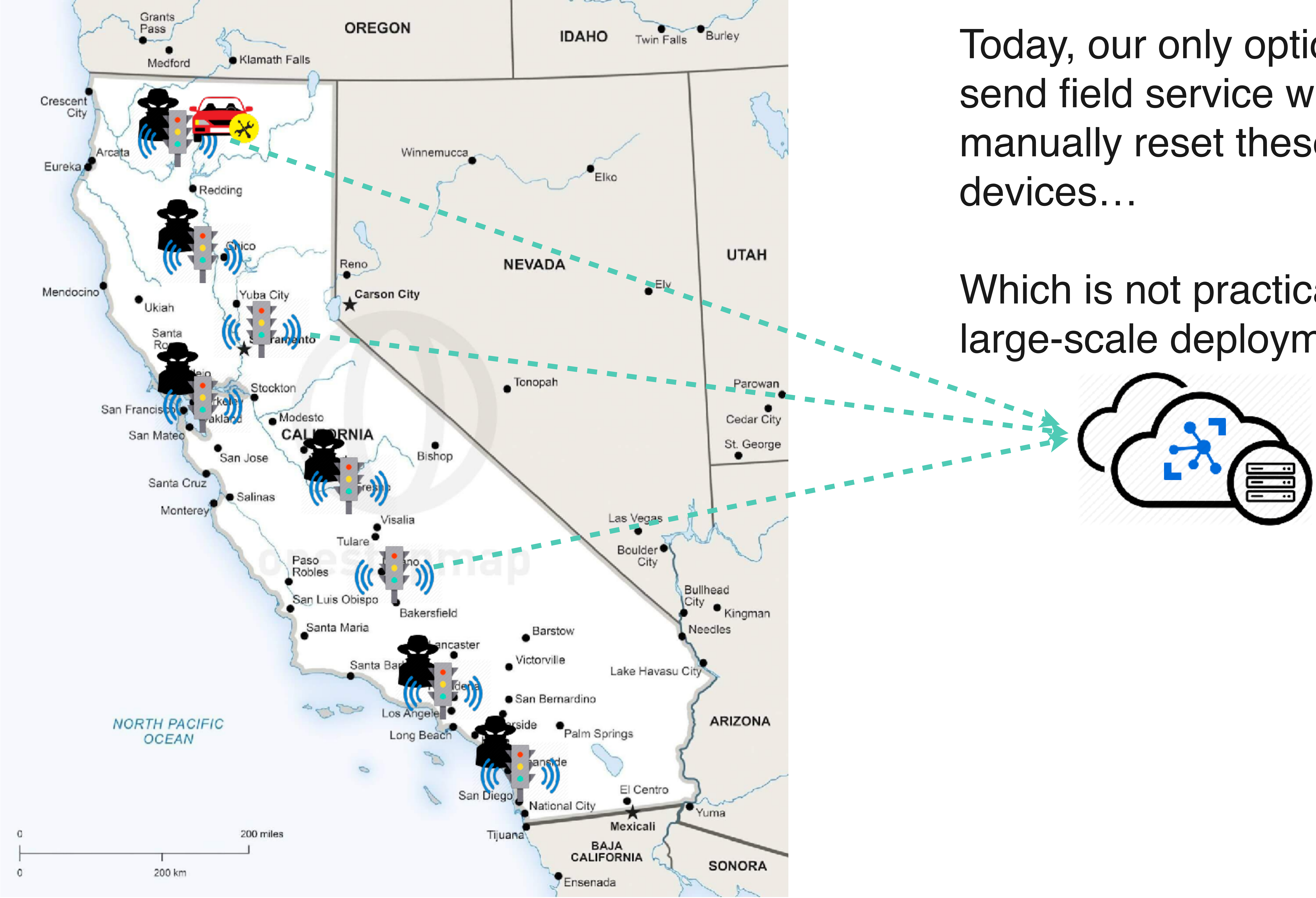
In normal cases these traffic lights send traffic condition reports to the IoT hub which replies with traffic policy.



But what if an attacker exploits a software vulnerability or a weak password?

Now all traffic lights in CA are controlled by a botnet.





Today, our only option is to send field service workers to manually reset these devices…

Which is not practical in such a large-scale deployment.

Can We Do Better?

Photo by Kate McGillivray @ CBC News 2017

Photo by Kate McGillivray @ CBC News 2017

Dominance in IoT

Definition: We say the hub dominates an IoT device if the hub can

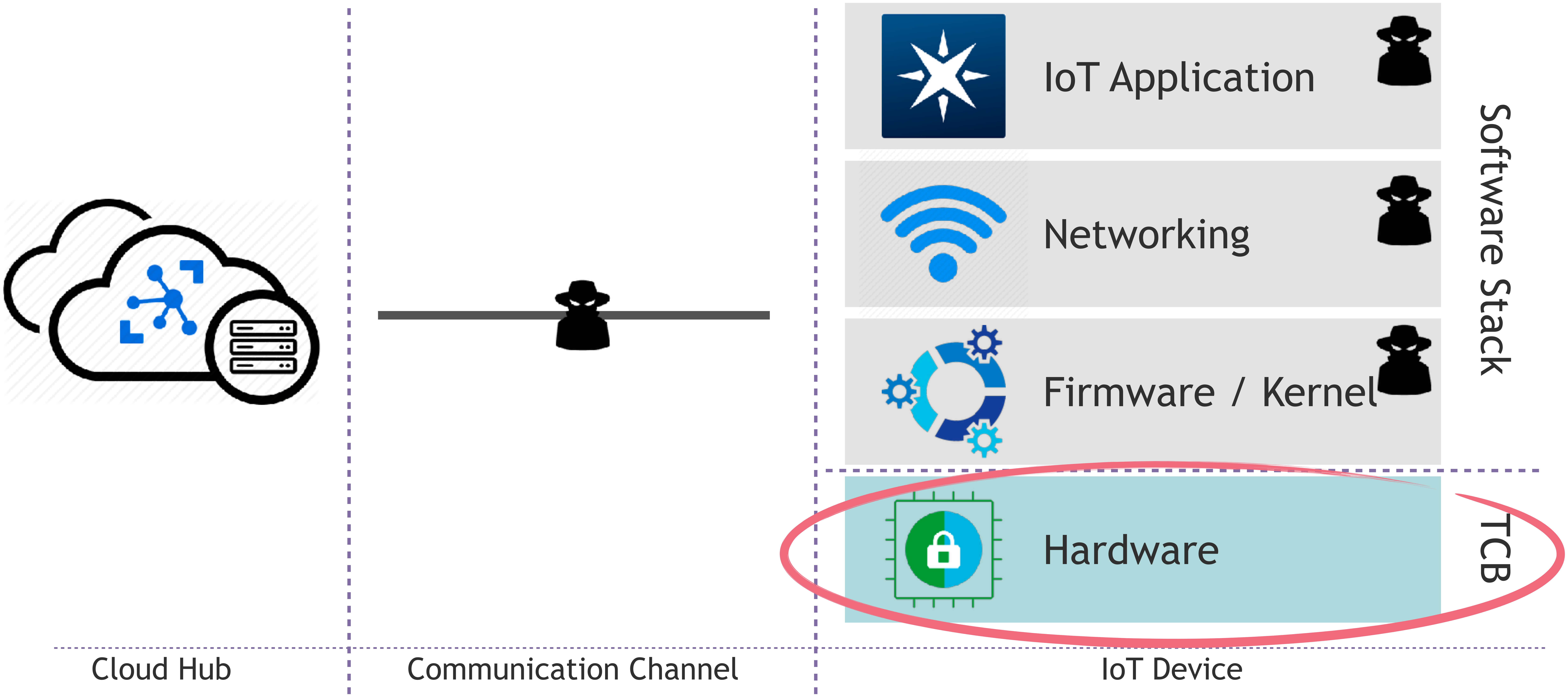
1. choose arbitrary code
2. force the device to run it within a bounded amount of time.

Dominance in IoT

Definition: We say the hub dominates an IoT device if the hub can

1. choose patched firmware
2. force the device to run it within four hours of attack discovery

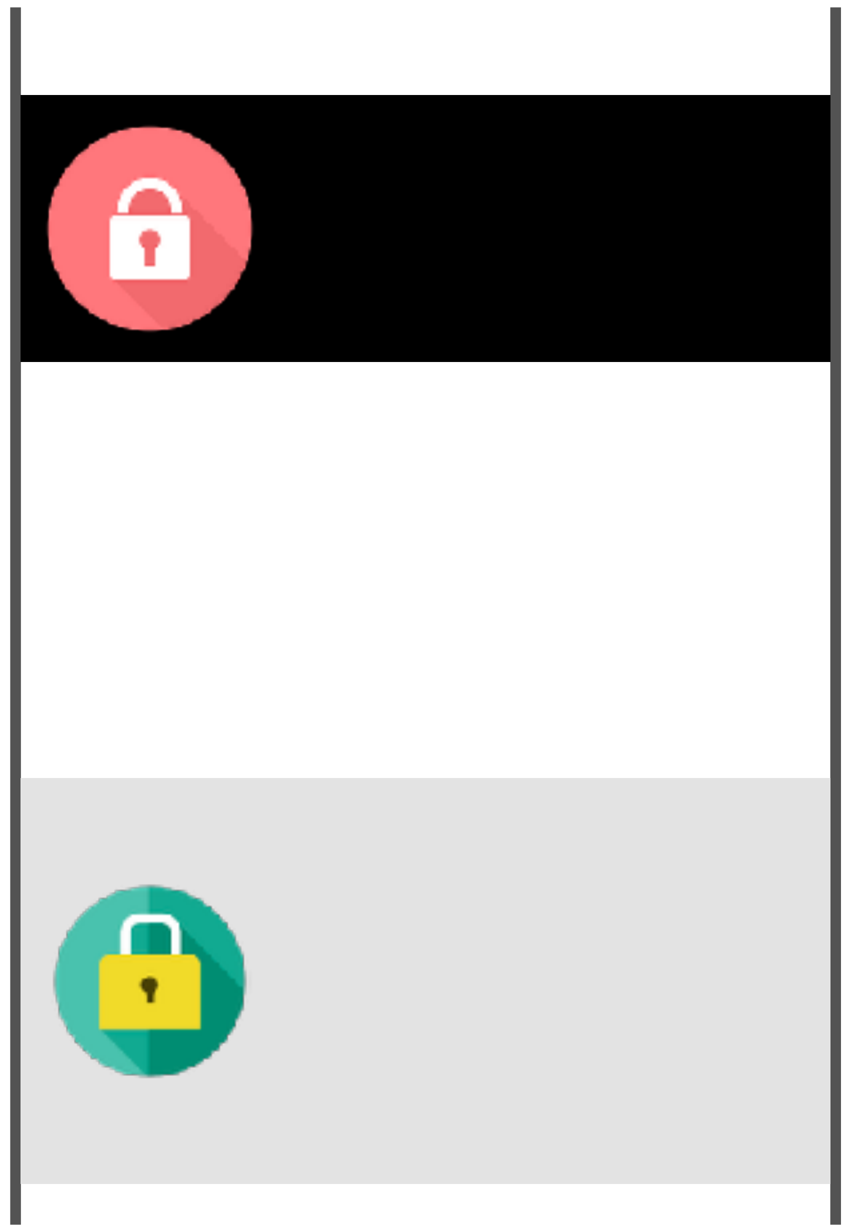
Dominance under Powerful Adversaries



Hardware Primitives

RWLatch: Read-Write Latch, blocks read and write to one or more storage regions until the next device reset

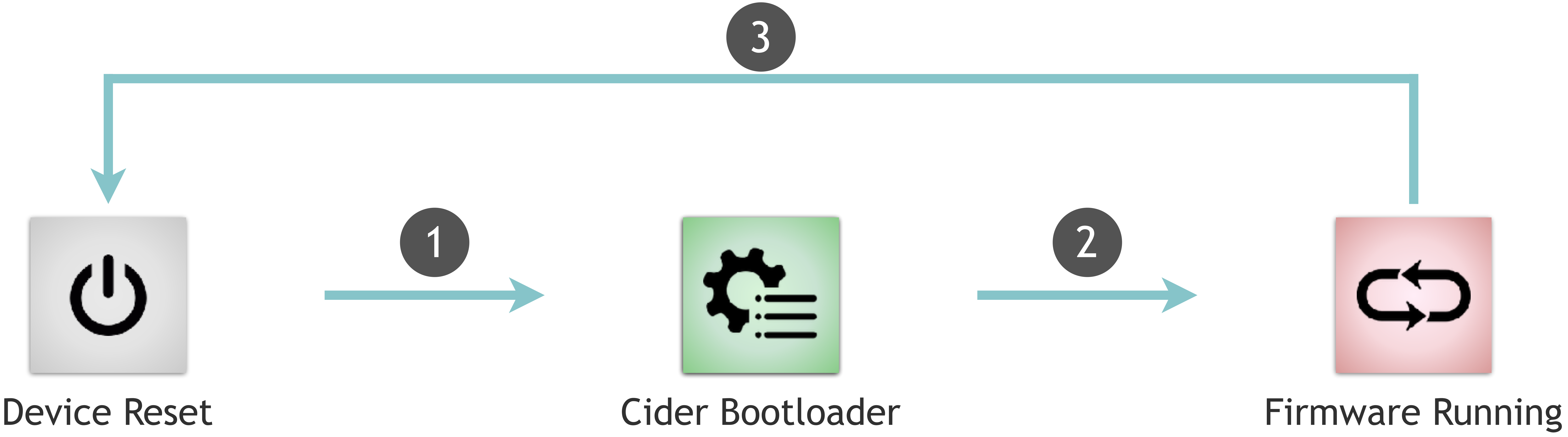
WRLatch: Write Latch, blocks write accesses to one or more storage regions until the next device reset



Storage

AWDT: Authenticated watchdog timer, a watchdog timer that is deferred only with certificates issued by the hub.

Get Dominance With Three Guarantees



Guarantee 1

Whenever the device is reset, it must boot into an unaltered Cider bootloader.

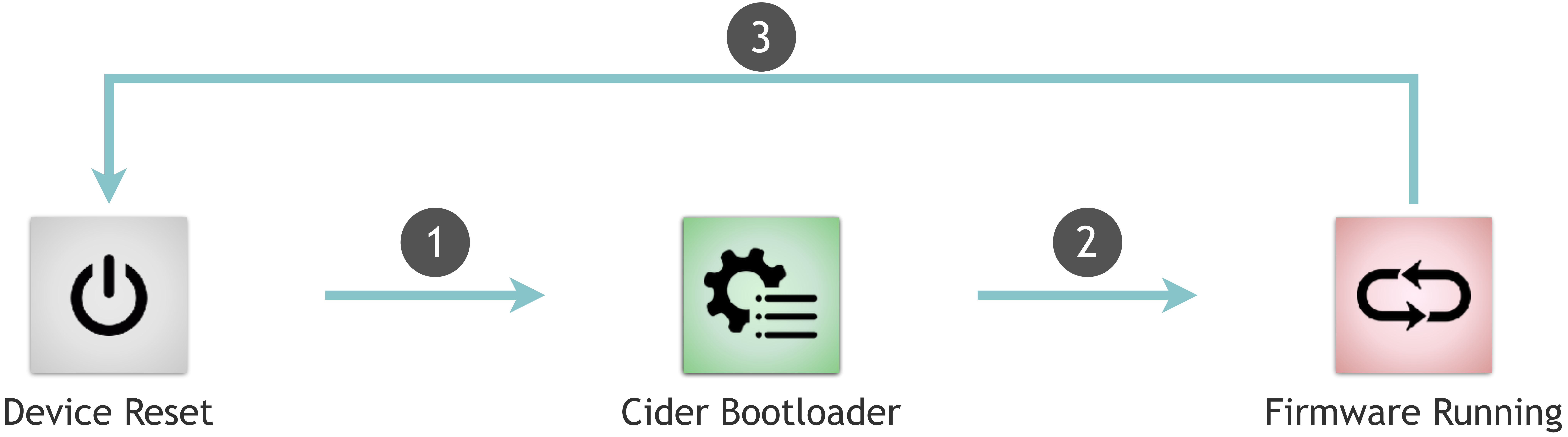
Guarantee 2

Cider bootloader transfers control to a firmware that is approved by the hub.

Guarantee 3

The hub may unconditionally force a device to reset within a time bound.

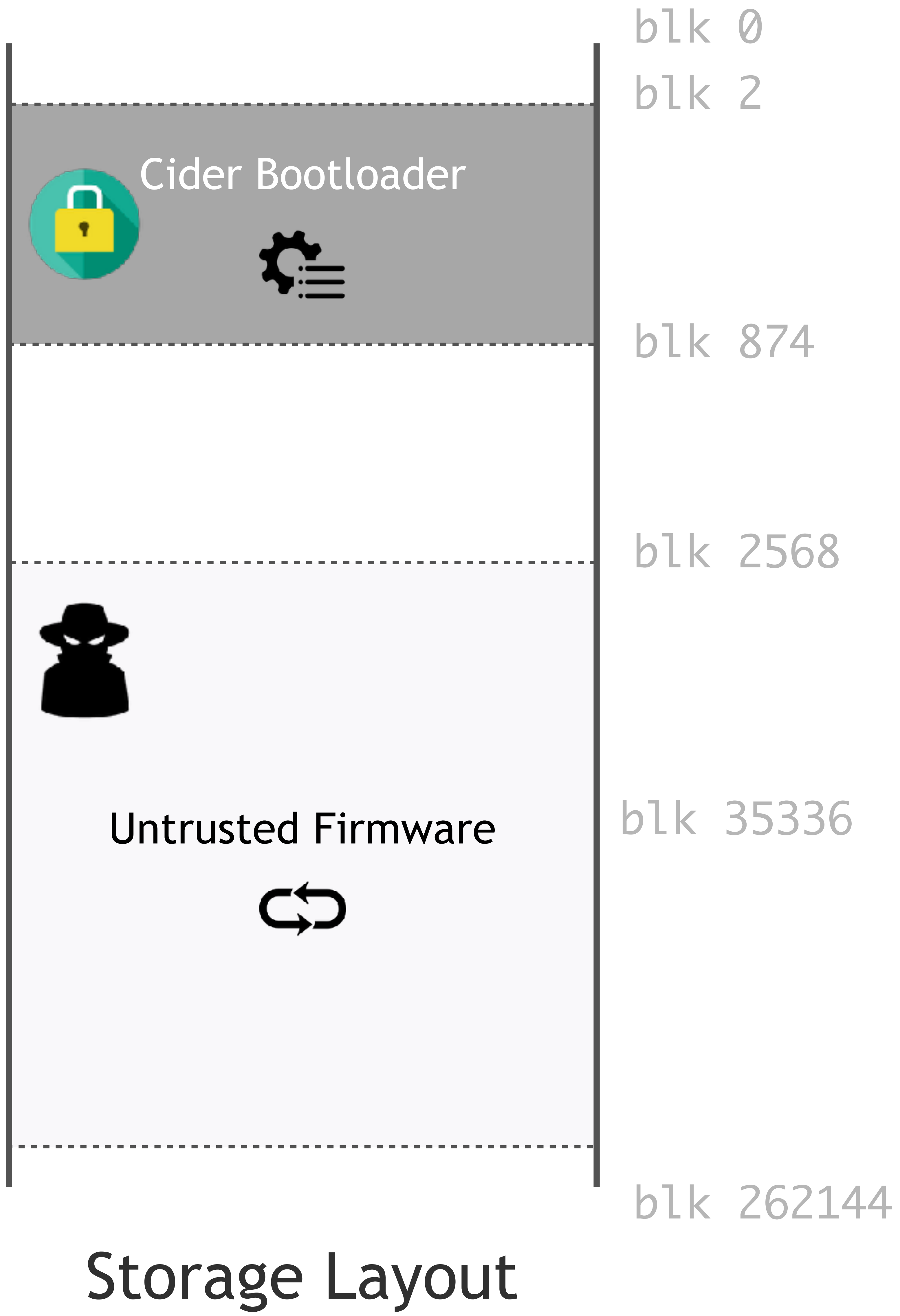
Isolation In Time



Isolation In Time: Alternating the execution of trusted and untrusted code in time.

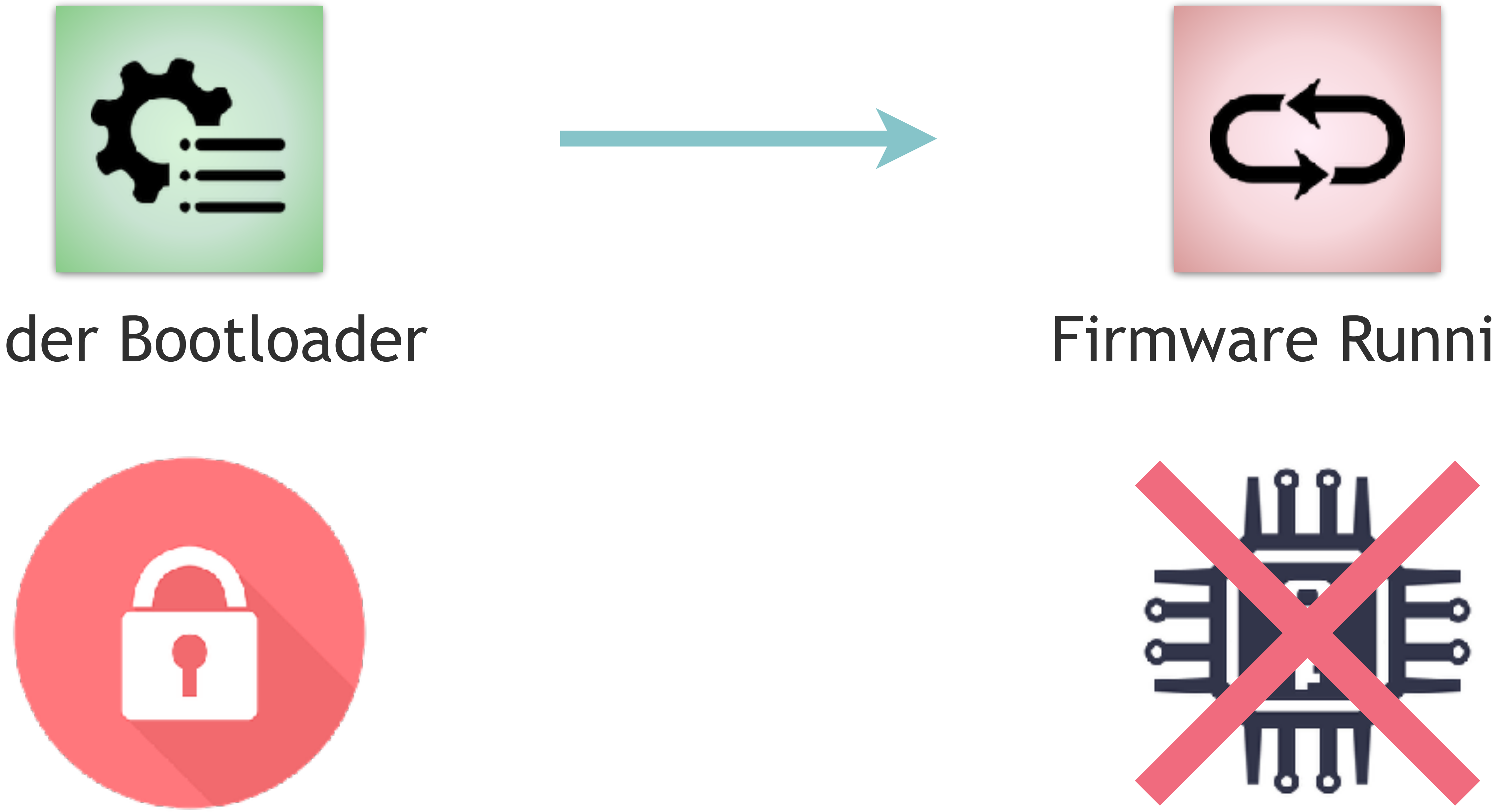
Guarantee 1: Reset into Unaltered Bootloader

WRLatch: Write Latch, blocks write accesses to one or more storage regions until the next device reset



Guarantee 2: Firmware Attestation & Patching

RWLatch: Read-Write Latch, blocks read and write to one or more storage regions until the next device reset



Cider Bootloader

Firmware Running

The attestation key is only consumed in Cider Bootloader

Guarantee 2: Firmware Attestation & Patching

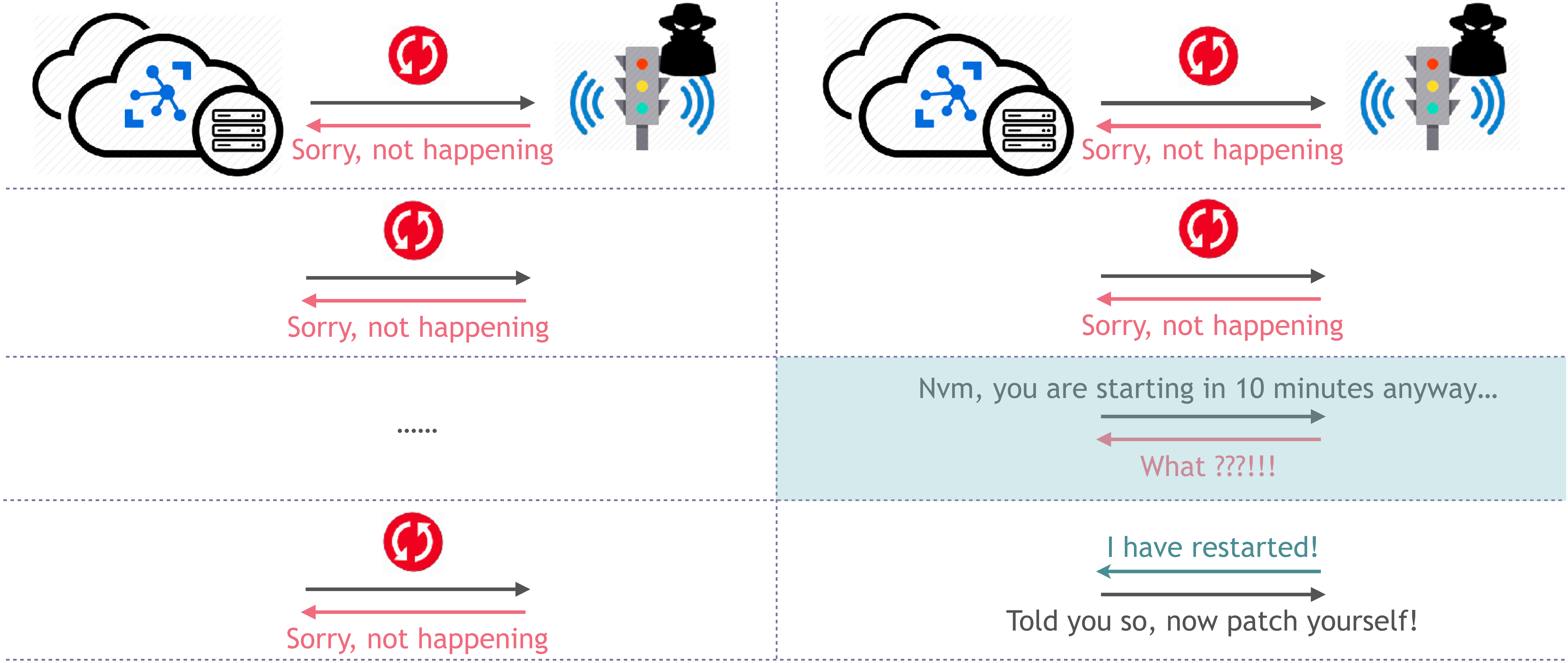
- **Networking Stack** is NOT part of our TCB.
  - Isolate the networking stack into a recovery module.
  - Treat the recovery module like the firmware, i.e., run it with all protections (RWLatch, WRLatch, AWDT) enabled.

Guarantee 2: Firmware Attestation & Patching

- **Networking** only when necessary (in our optimized scheme).
  - In normal circumstances when the firmware is cooperating, Cider does not involve boot-time networking.
  - Firmware attestation and patching is required only when the hub is questioning the device firmware integrity.

For details, please refer to our paper.

Guarantee 3: Hub-Enforced Unconditional Reset



Sorry, not happening

Sorry, not happening

……

Sorry, not happening

Once Rooted, Forever Rooted

Sorry, not happening

Sorry, not happening

Nvm, you are starting in 10 minutes anyway…

What ???!!!

I have restarted!

Told you so, now patch yourself!

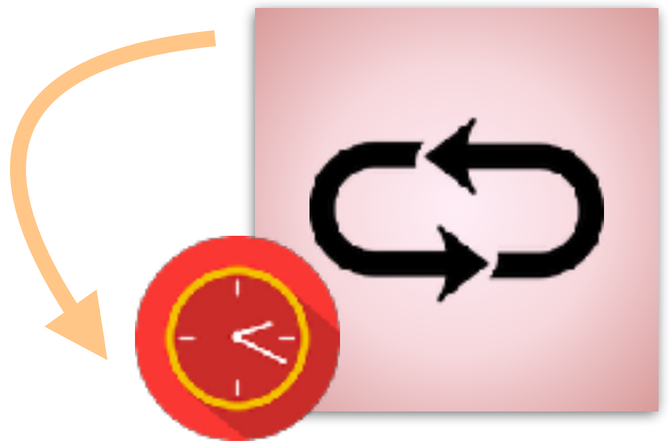
Rooted and Recovered with Cider

Trial: Conventional Watchdog Timer (WDT)

- **Popular** among IoT devices
- **Reliability Guarantee** against buggy IoT firmware that hangs occasionally

Trial: Conventional Watchdog Timer (WDT)

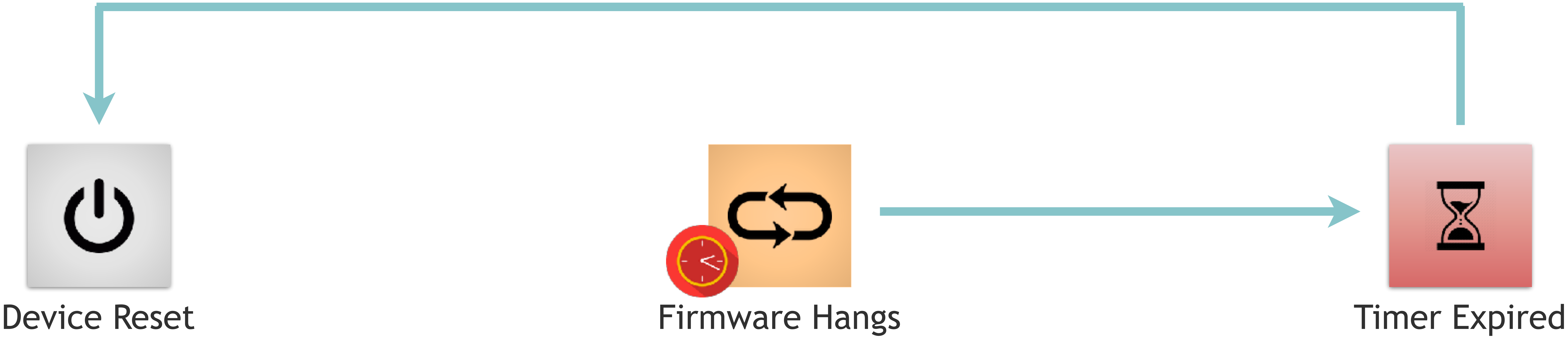
```
MOV REG1, 0xdeadbeef
```



Firmware Running

0:05

Trial: Conventional Watchdog Timer (WDT)



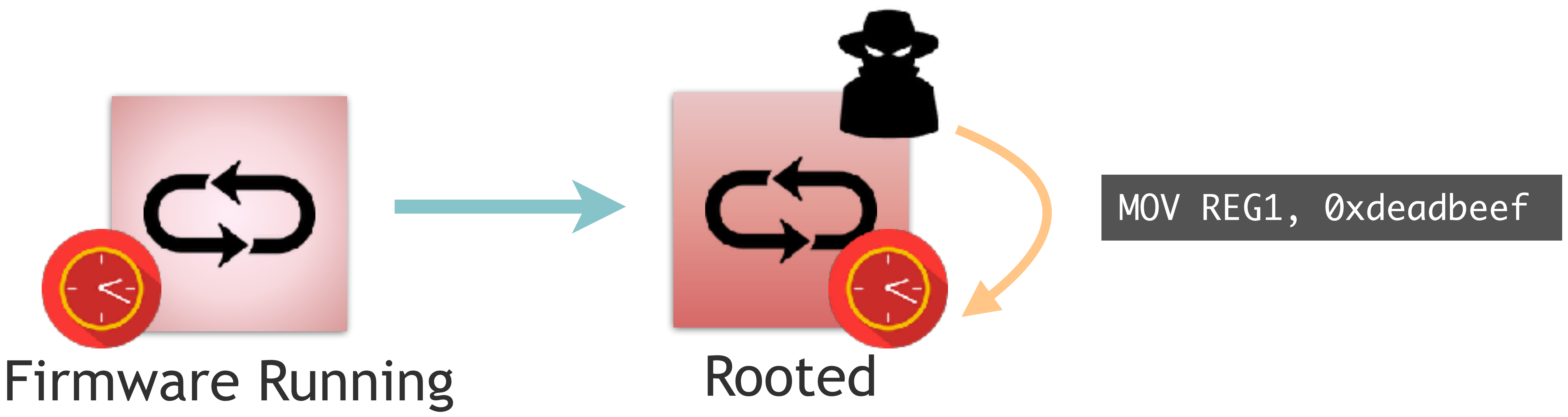
Device Reset

Firmware Hangs

Timer Expired

0:00

Security Issue of WDT



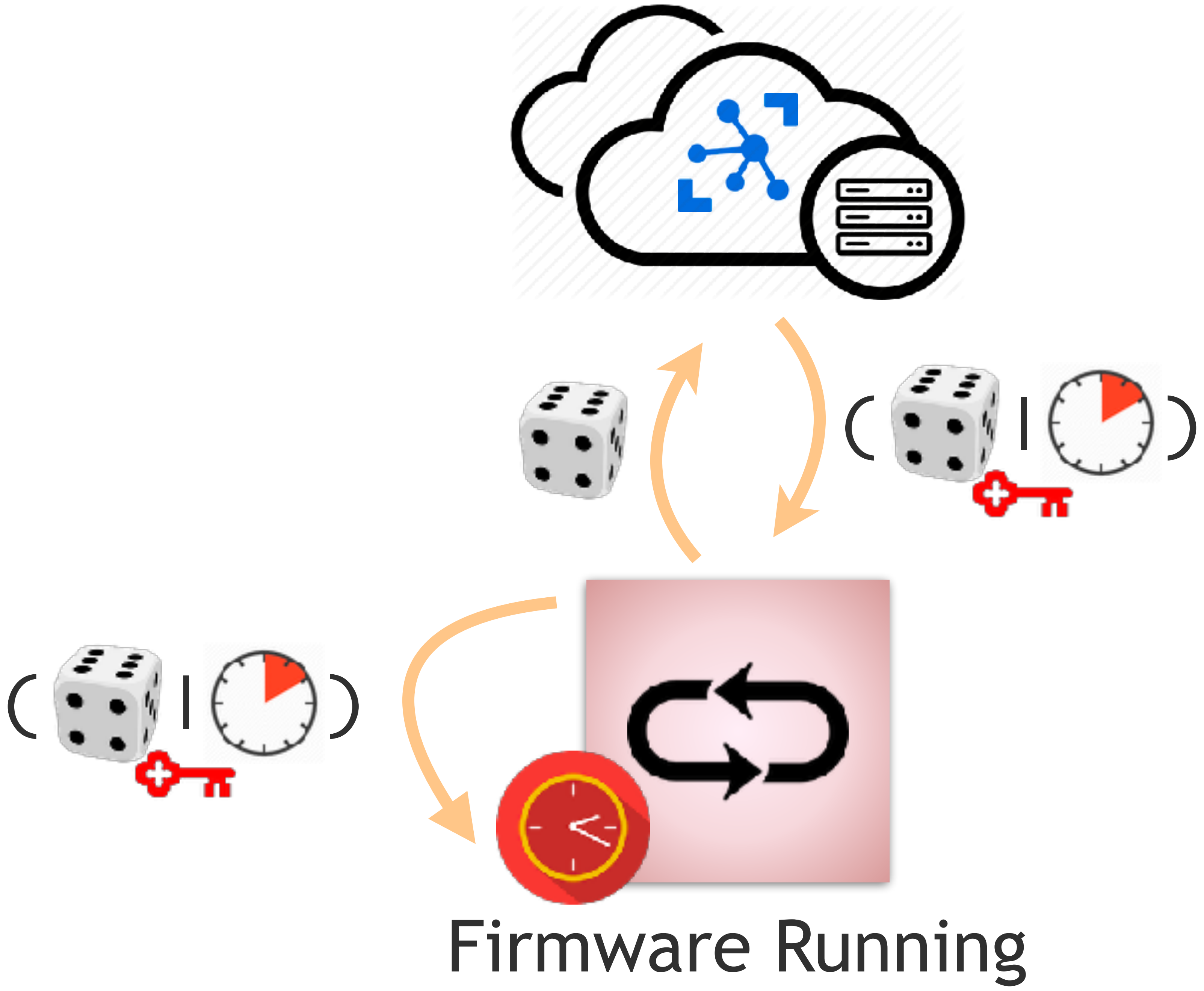
```
MOV REG1, 0xdeadbeef
```

0:05

Security Issue

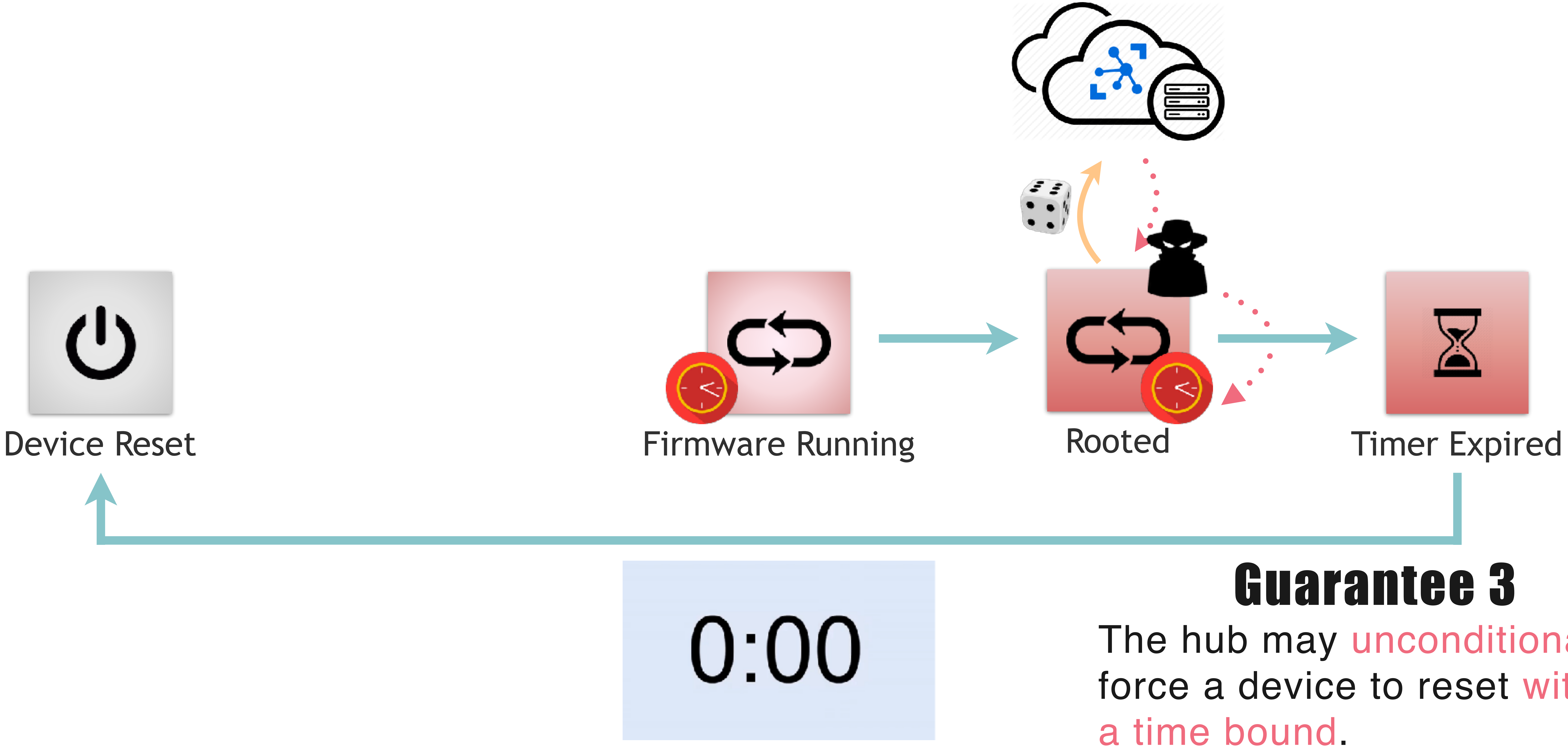
Conventional watchdog timer can be serviced by attacker as well given it has full control over the firmware.

Solution: Authenticated Watchdog Timer



0:05

Solution: Authenticated Watchdog Timer



Guarantee 3

The hub may unconditionally force a device to reset within a time bound.

Implementing Authenticated Watchdog Timer

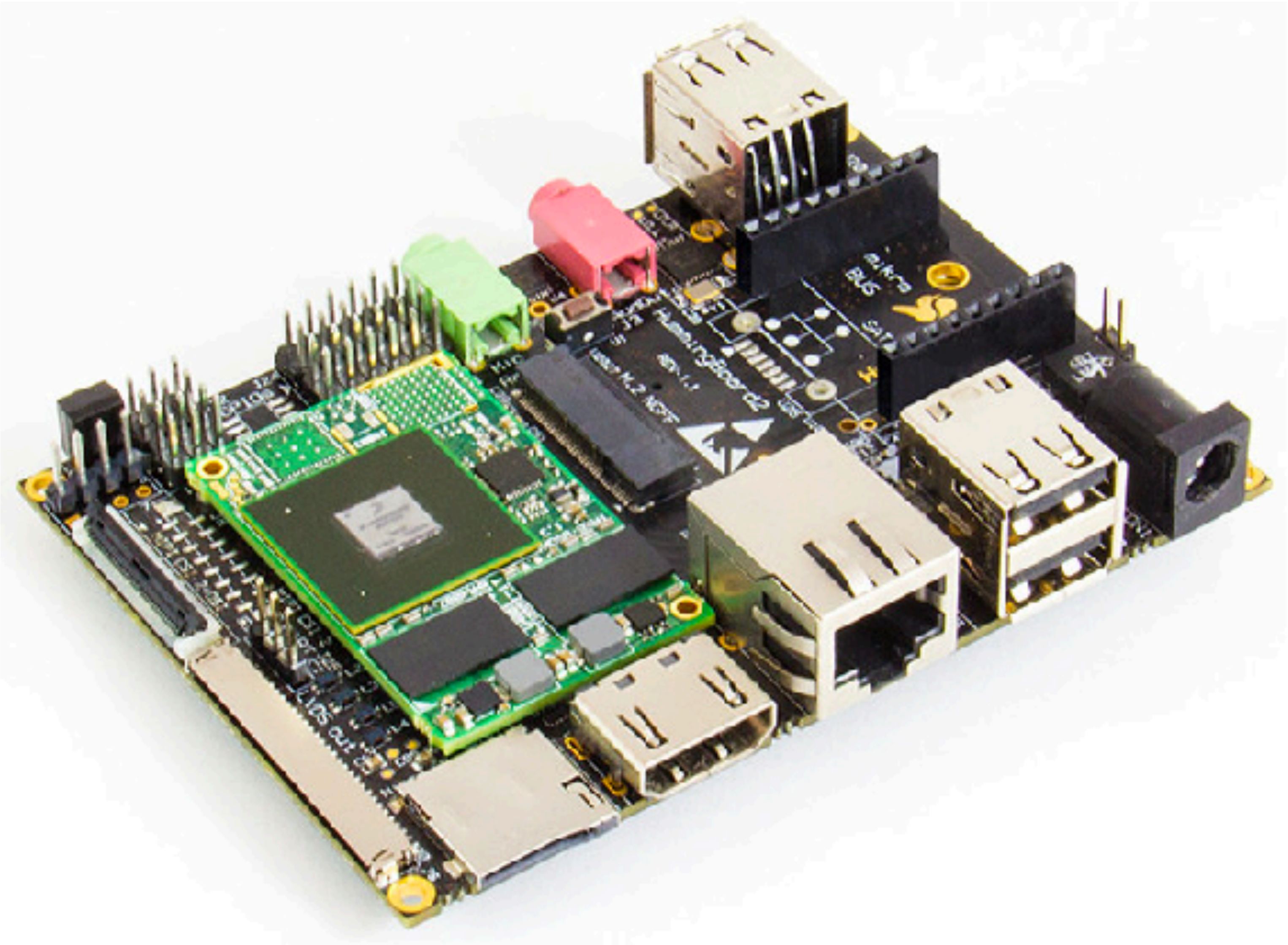
- **New Concept**, no commodity AWDT hardware available

- **eAWDT**: Attach an external AWDT built out of MCU
  - STM32L053R8 (cost < $3)
  - ATECC608A + ATtiny412 (cost < $1)

- **Repurpose** existing hardware
  - TrustZone
  - BCM Secure Physical Timer
  - Memory Protection Unit

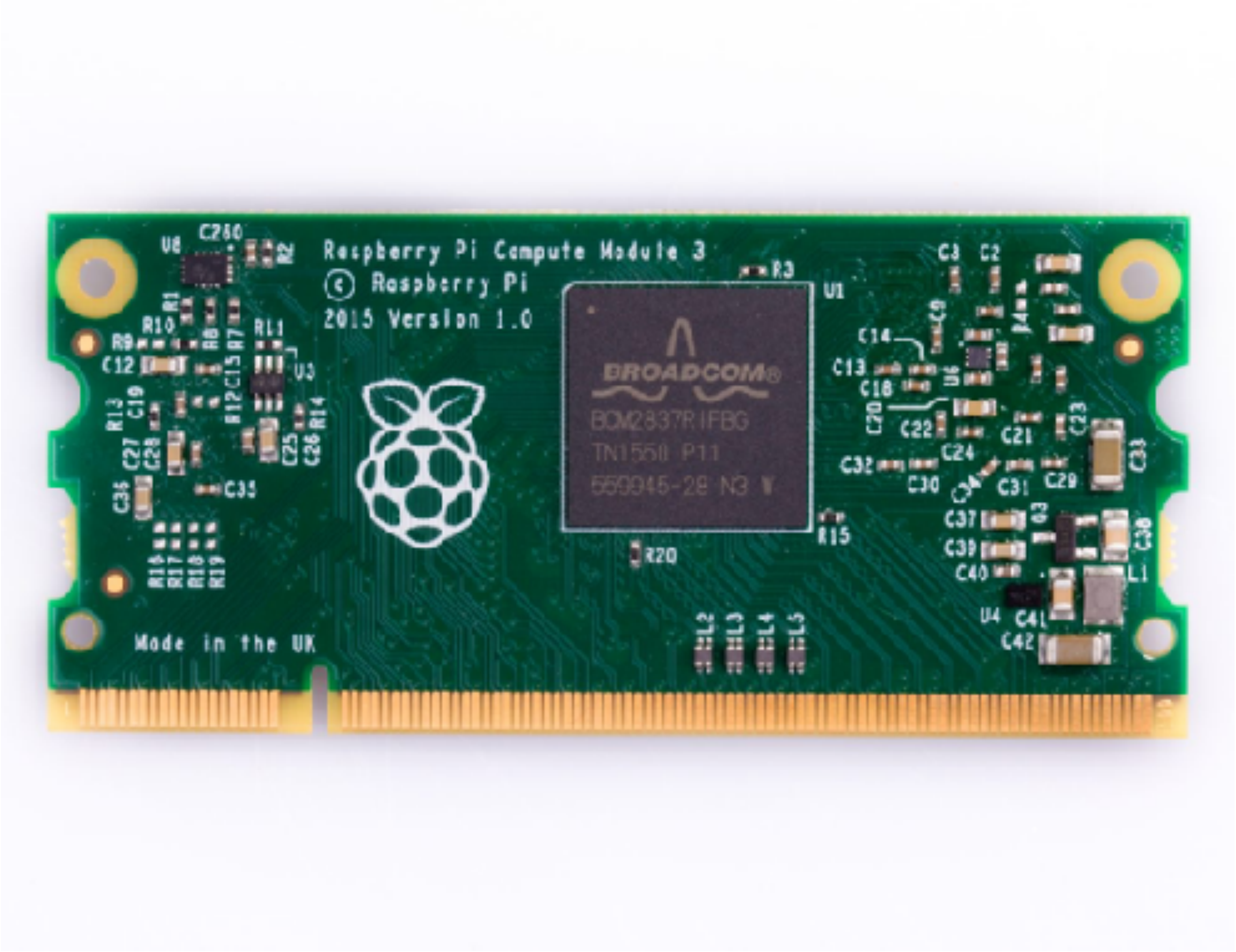
For details, please refer to our paper.

Prototypes



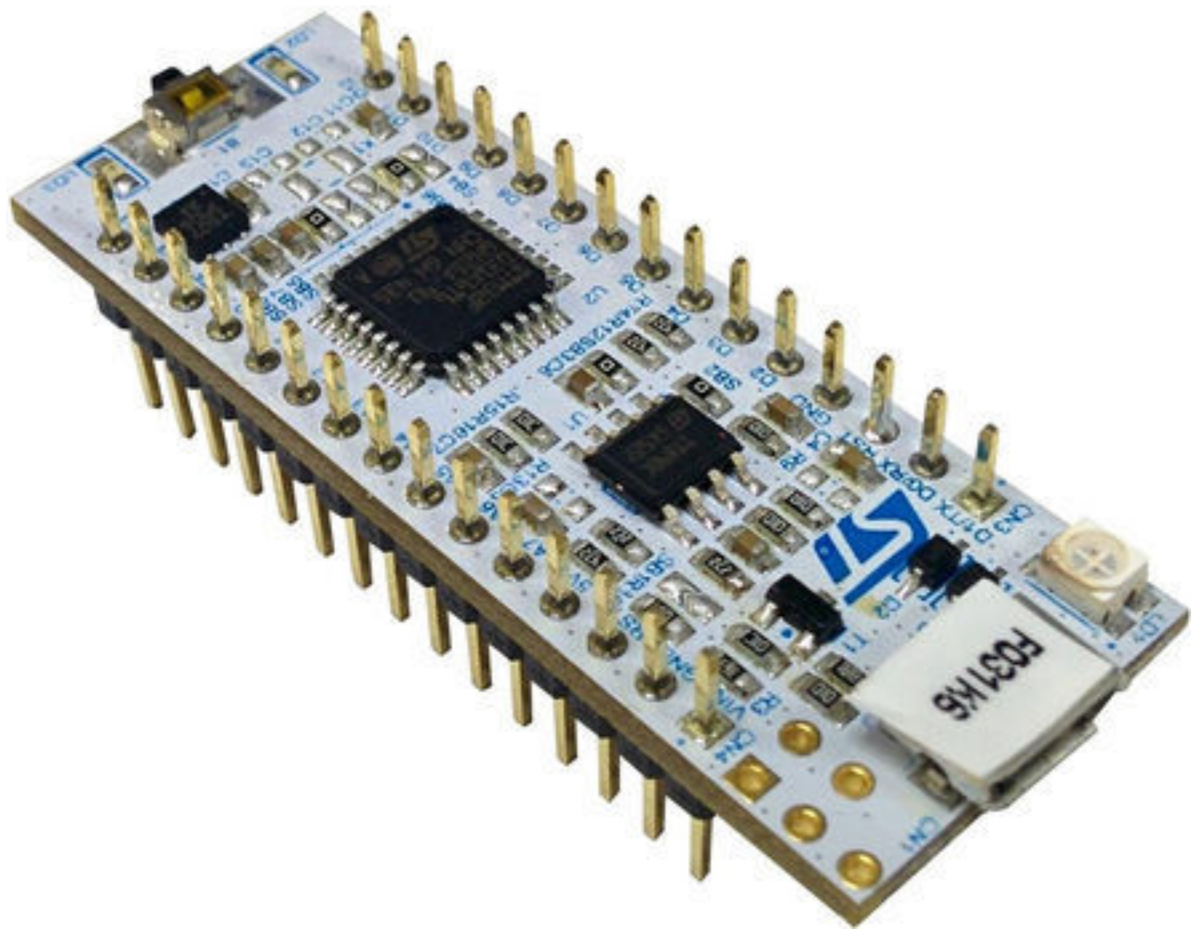
SolidRun
HummingBoard Edge
(HBE)

$240



Raspberry Pi
Compute Module 3
(CM3)

$120



STMicroelectronics
Nucleo-L476RG
(NL476RG)

$15

Prototypes

	WRLatch	RWLatch	AWDT	eAWDT
SolidRun HummingBoard Edge	eMMC power-on write protection	Built-in CAAM Crypto Module	TrustZone	*External AWDT*
Raspberry Pi Compute Module 3	eMMC power-on write protection	*OPTIGA SLB 9670 (Any TPM 2.0 chip)*	SPT + EL3	
STMicroelectronics Nucleo-L476RG	MPU Firewall	MPU Firewall	MPU + IWDG	

Summary: The hardware primitives are mostly available on the three IoT boards. For those that are not available, they can be obtained and plugged into the board easily with low cost.

Evaluation: Software Compatibility

Device	Firmware	Compatible
HBE	Windows IoT Core	✓
	Debian	✓
CM3	Raspbian	✓
	Buildroot	✓
NL476RG	FFT (Bare-metal app)	✓
	TLC (Bare-metal app)	✓

Summary: Cider is compatible with common firmware and bare-metal applications that run on the tested boards.

Evaluation: Performance - Boot Time

Config	HBE		CM3		NL476RG	
Baseline (w/o Cider)	0.98		1.25		0.01	
Normal case (w/ Cider)	1.25	+0.27	1.73	+0.48	4.35	+4.34
Attestation & Patching	15.60	+14.60	20.80	+19.50	30.20	+30.20

Summary: The additional boot time under normal circumstances is spent on firmware integrity checking. In the case of attestation and patching, the boot time is affected by the size of the patch.

Evaluation: Performance - Runtime Delay

Config	HBE		CM3		NL476RG	
1min Fetching Interval	0.28%	± 0.54%	0.32%	± 0.97%	0.64%	± 0.30%
5min Fetching Interval	0.15%	± 0.53%	0.09%	± 0.58%	0.16%	± 0.26%

Summary: Cider (ticket fetching) incurs negligible runtime overhead.

Discussion: Minimal Requirements on Hardware

Provide a solution that is not only simple in software complexity, but more importantly, requires ***a minimal hardware TCB***.

Discussion: Minimal Requirements on Hardware

Runtime Isolation	Isolation in Time
Multi-threading (CPU slicing, TLB flushes, etc)	Latches (RWLatch, WRLatch)
Ring-0/1/2/3, privilege levels (as a social norm)	
Page tables, Memory Management Units (MMU)	Authenticated Watchdog Timer
Interrupts, context switches	

Vulnerable to side-channels, spectre, …, many types of attacks on hardware (lessons learned from Day 1 Session 1)

Simplicity is the key: Cider is immune to speculative execution and common side-channel attacks and is perfect for providing a security cornerstone for IoT.

Conclusion

- **Dominance** is necessary in the presence of large-scale industrial IoT deployments: *we need to return thousands of devices to their original missions after being compromised.*

- **Cider** is a practical scheme that enforces dominance on IoT devices via three guarantees: boot to Cider, firmware attestation & patching, unconditional reset.

- **Evaluation** shows that Cider is compatible with a wide range of IoT boards and firmware while introducing negligible overhead.

Q & A

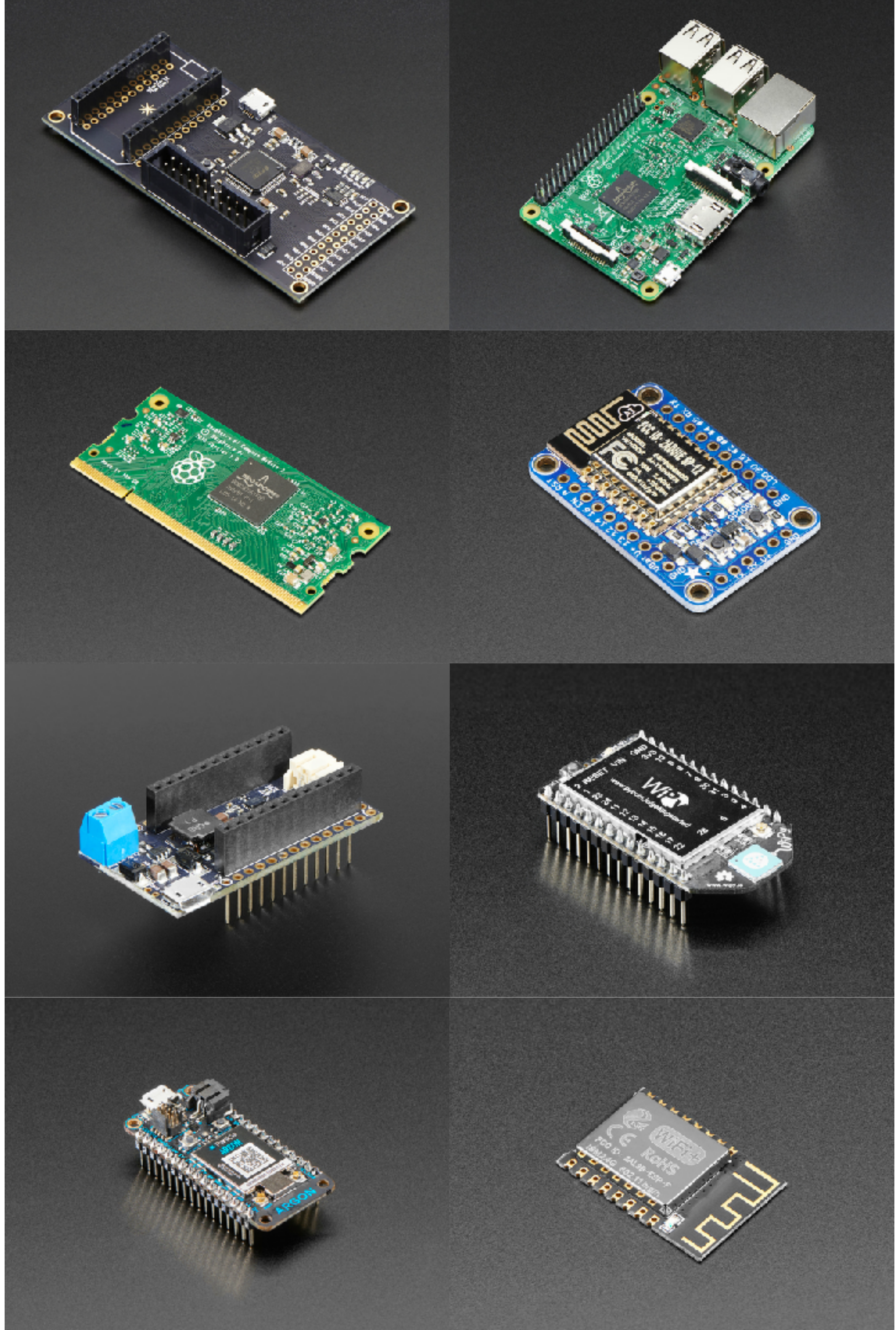
Why not Using IPMI ?

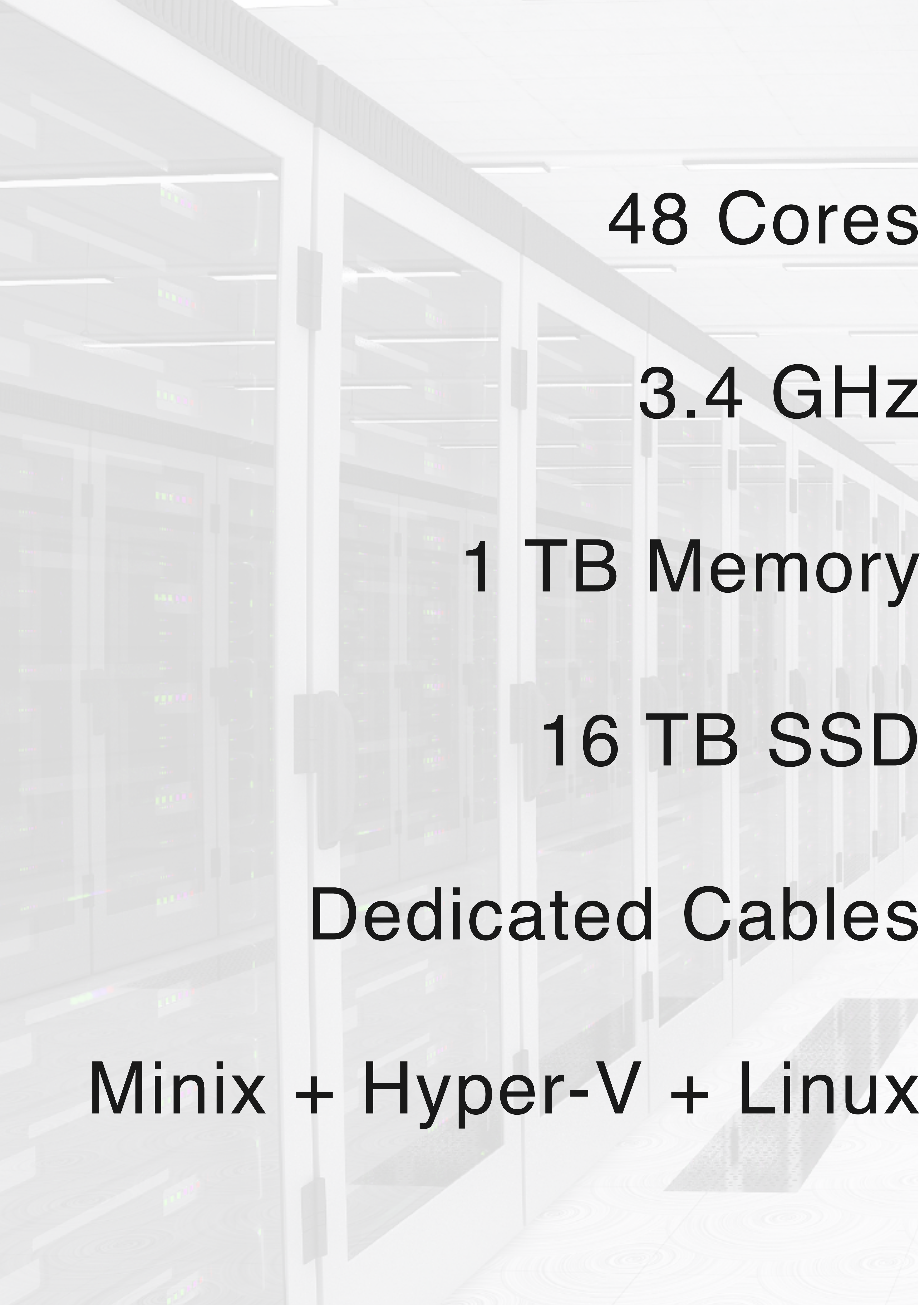
Q: How to update thousands of machines in a data center?

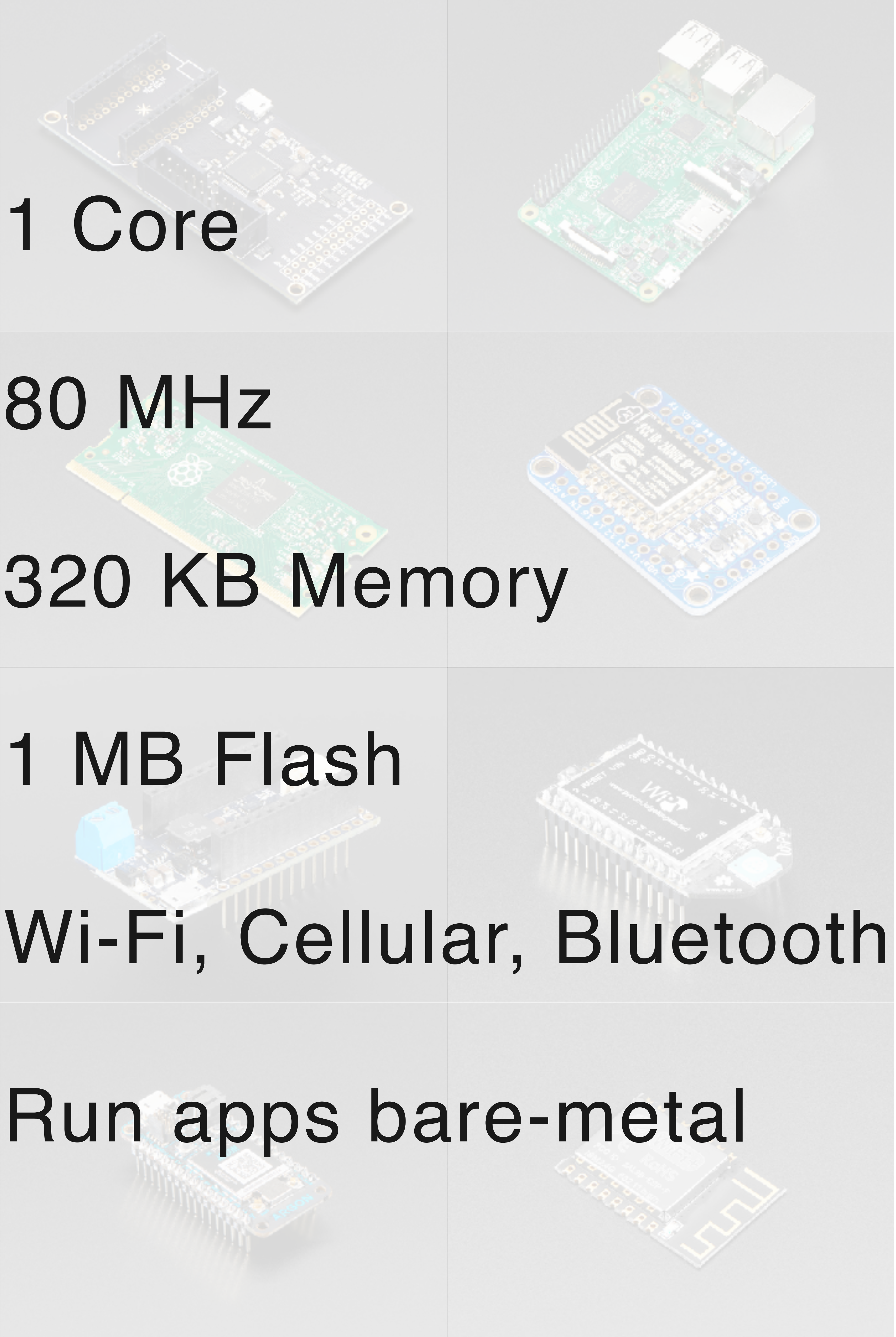
A: Haven’t you heard about the magical Intelligent Platform Management Interface? They even run Minix OS in it!

A: Even if IPMI fails, I can still take the disk out, reformat it, install the patched software, and clear out the malware.

VS







48 Cores	VS	1 Core
3.4 GHz		80 MHz
1 TB Memory		320 KB Memory
16 TB SSD		1 MB Flash
Dedicated Cables		Wi-Fi, Cellular, Bluetooth
Minix + Hyper-V + Linux		Run apps bare-metal

Existing solutions like IPMI are not suitable for resource-constrained IoT devices

What If The Networking Stack Gets Hacked?

What If The Networking Stack Gets Hacked?

- **Worst Case**: Cider bootloader gets into infinite loop → DoS
    - Seek help from ISPs to temporarily block attacker’s traffic until Cider updates itself.

What If The Networking Stack Gets Hacked?

- **Limited Attacking Surface**:
    - Cider always initiates connections actively.
        - Cider never has open ports waiting for incoming instructions.
    - Cider only connects to the hub via either hardcoded information
        - domain names or IP addresses.