



Making Device Identity Trustworthy

Matthew Garrett
Brandon Weeks

Open Source Summit Europe 2019
October 28, 2019

About us

Matthew Garrett
Low-level security
 @mjg59



Brandon Weeks
High-level security
 @__weeks__



Agenda

1

Device identity

2

Trusted Platform Modules

3

Binding TPM and device identity

4

Putting it all together

5

Open source tooling

Device identity

Identity for remote access (BeyondCorp)



The Fundamental Problem

How do we determine which device we're talking to?



Devices provisioned with **known image and identity**



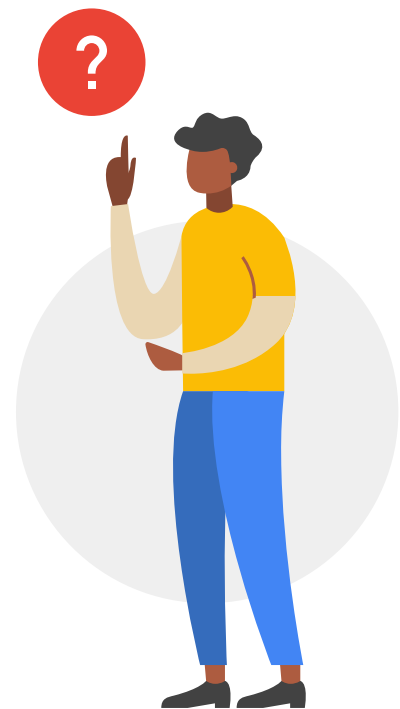
Issued to users



Users **access resources** from diverse environments with varying risk

What is (secure) device identity?

- Unique
 - Device serial number, asset tag
- Bound to hardware
 - Identifies the device hardware itself
 - Stable for the lifetime of the hardware
- Prevents impersonation
 - Unforgeable
 - Resistant to key-compromise impersonation attacks



Existing solutions are inadequate

- **Self identification** can be trivially forged
- **On disk identity keys** can be duplicated
- **Trust bootstrapping** is difficult and risky



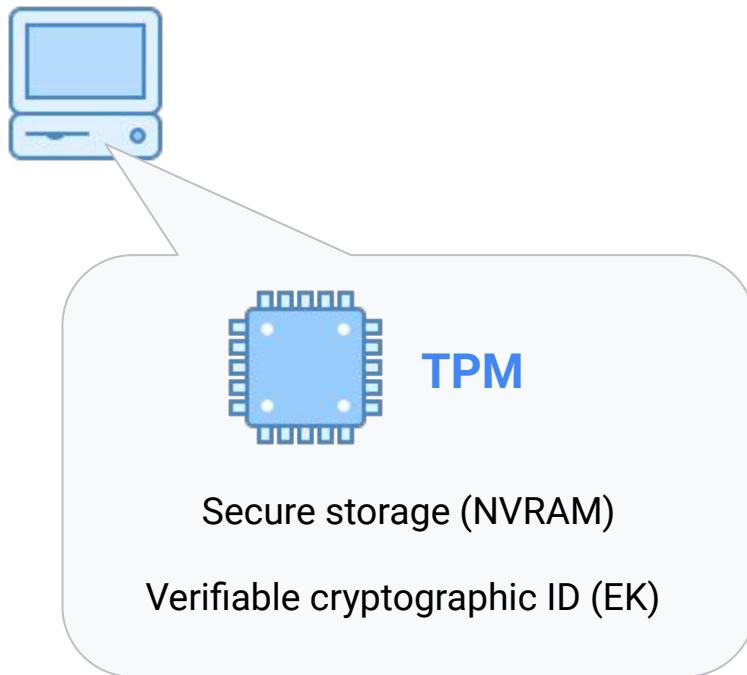
Trusted Platform Modules

TPM Overview

If we can't trust an operating system, what can we trust?

Trusted Platform Modules allow us to segregate that trust into a separate hardware component

- Store a trusted record of what the system booted
- Securely store keys and/or secrets
- Has a unique cryptographic identity



TPM Identity: Endorsement Keys and Certs

- Each TPM has one or more unique Endorsement Keys
- Each EK has an associated certificate
- The certificate can be chained back to the TPM manufacturer



Endorsement Key

RSA or ECC key pair

The Endorsement Key uniquely identifies the TPM



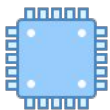
Endorsement Certificate

<Endorsement key info>
Issuer information: ...
Signed by: ...
...

The Endorsement Certificate relates the TPM to its manufacturer

Attestation Keys

- AKs allow attestation to be separated from the EK
- Through a complicated dance, the TPM with a specific EK can prove it generated a AK
- The AK can then in turn be used to certify other TPM-generated keys



TPM proves AK is
co-resident with EK



AK signs
TPM-generated key



Proof of key possession
transmitted

Binding TPM and Device identity

TPMs aren't Devices

- EK certificates prove TPM identity
- EK certificates don't prove device identity



Binding at runtime

We can simply ask each machine to report its EK certificate and existing identity

Upsides: involves no new procedures or infrastructure

Downsides: depends on systems all being in a trustworthy state



Script extracts TPM EK certificate and device information

This is placed in an **internal database**

The **EK certificate** can be used as a lookup to verify device identity

Binding at provisioning

TPM identity and machine identity can be bound when a machine is initially provisioned

This reduces the window where a system could be compromised, but still relies on a trustworthy human who never makes mistakes



IT Officer extracts TPM EK and device information

This is placed in an **internal database**

The **EK** can be used as a lookup to verify device identity

Binding at manufacture

**The device manufacturer knows
both the machine and TPM
identities**

If the device manufacturer can provide cryptographic evidence of this binding, the identities are bound even before the machine is shipped.

Platform Certificates

Platform Certificates bridge the gap between the device and the TPM

These are X.509 Attribute Certificates (RFC 5755) describing additional attributes that can be associated with the EK. Examples include:

- Device manufacturer
- Device model
- Device serial number



Device manufacturer extracts TPM EK and device information



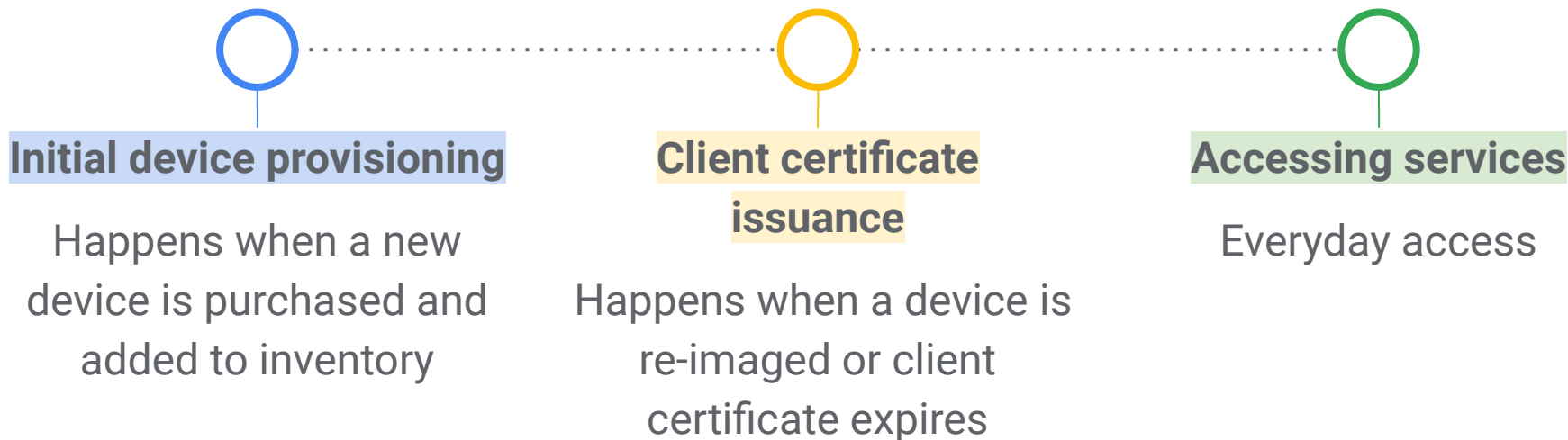
Platform certificate containing this information is generated



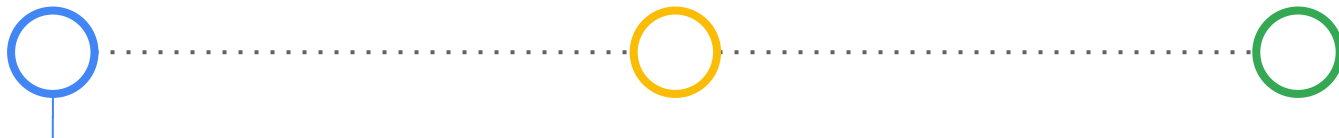
Platform certificate is provided to **device owners**, allowing automatic enrollment

Putting it all together

Putting it all together



Putting it all together



Initial device provisioning

Happens when a new device is purchased and added to inventory

- An Attestation Key is generated by the TPM
- Attestation Key and Endorsement Key Certificate are presented to the Attestation CA
- Attestation CA issues an Attestation Key Certificate

Putting it all together



Client certificate issuance

Happens when a device is re-imaged or client certificate expires

- A signing key is generated by the TPM
- A “proof” of the key properties is signed by the Attestation Key
- The proof and Attestation Key Certificate can be exchanged for an X.509 client certificate

Putting it all together



- The client certificate is used with mutual TLS authentication to the access gateway

Now what?

Device identity isn't just about clients

We already have multiple implementations of host identity

- TLS certificates
- SSH host keys

Why are we bootstrapping trust more than once?



Intermediate certificate issued, chaining back to institute root cert



Service keys can be generated on the TPM, chaining back to the intermediate



Clients can **automatically** obtain proof of device identity

Open source tooling

Go-attestation

Available at:

<https://github.com/google/go-attestation>

Implements:

- All TPM primitives for an attestation client on Windows/Linux, TPM 1.2 or TPM 2.0.
- Platform Certificate support merged today

```
go-attestation: github.com/google/go-attestation/attest      Index | Files | Directories

package attest

import "github.com/google/go-attestation/attest"

Package attest abstracts TPM attestation operations.

Index

Variables
func ParseEKCertificate(ekCert []byte) (*x509.Certificate, error)
type AIK
    func (k *AIK) ActivateCredential(tpm *TPM, in EncryptedCredential) (secret []byte, err error)
    func (k *AIK) AttestationParameters() AttestationParameters
    func (k *AIK) Close(t *TPM) error
    func (k *AIK) Marshal() ([]byte, error)
    func (k *AIK) Quote(tpm *TPM, nonce []byte, alg HashAlg) (*Quote, error)
type AIKPublic
    func ParseAIKPublic(version TPMVersion, public []byte) (*AIKPublic, error)
type ActivationParameters
    func (p *ActivationParameters) Generate() (secret []byte, ec *EncryptedCredential, err error)
```

Questions?

