



# Quo Vadis, certificates?



**Matteo Bolognini**  
Product Specialist  
[skartek.dev](http://skartek.dev)

# Agenda

- ▶ Understanding certificates
- ▶ Usage of certificates
- ▶ SCEP
- ▶ ACME

# Understanding certificates

- ▶ A certificate is essentially an electronic "passport" that uses cryptographic techniques to prove the identity of the holder.
- ▶ It contains a **public key**, which is part of a cryptographic key pair, along with information about the key's owner (like their name, organization, and the certificate's validity period).
- ▶ The certificate is signed by a **Certificate Authority (CA)**, which is a trusted entity in the PKI.

# Understanding certificates

- ▶ **Subject:** The entity (individual, server, etc.) to which the certificate belongs

# Understanding certificates



The image shows a certificate chain and its details. The chain is as follows:

- Amazon Root CA 1
- └ Amazon RSA 2048 M02
- └ \*jamfcloud.com

For the certificate **\*jamfcloud.com**:

- Issued by: Amazon RSA 2048 M02
- Expires: Saturday 1 February 2025 at 23:59:59 Greenwich Mean Time
- This certificate is valid

**Details**

Subject Name	
Common Name	*jamfcloud.com

**Issuer Name**

**Country or Region** US

**Organisation** Amazon

**Common Name** Amazon RSA 2048 M02

# Understanding certificates

- ▶ **Subject:** The entity (individual, server, etc.) to which the certificate belongs
- ▶ **Serial Number:** A unique number identifying the certificate

# Understanding certificates



Amazon Root CA 1

└ Amazon RSA 2048 M02

  └ \*.jamfcloud.com

**Country or Region** US

**Organisation** Amazon

**Common Name** Amazon RSA 2048 M02

**Serial Number** 01 3D 9A 3F A1 3F 6C 72 93 8F 7A 5C 17 2A E0 88

**version** 3

**Signature Algorithm** SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 )

**Parameters** None

**Not Valid Before** Wednesday 3 January 2024 at 00:00:00 Greenwich Mean Time

**Not Valid After** Saturday 1 February 2025 at 23:59:59 Greenwich Mean Time

**Public Key Info**

**Algorithm** RSA Encryption ( 1.2.840.113549.1.1.1 )

# Understanding certificates

- ▶ **Subject:** The entity (individual, server, etc.) to which the certificate belongs
- ▶ **Serial Number:** A unique number identifying the certificate
- ▶ **Validity Period:** The time frame during which the certificate is considered valid

# Understanding certificates



 Amazon Root CA 1
└  Amazon RSA 2048 M02
└  *jamfcloud.com
<b>Country or Region</b> US
<b>Organisation</b> Amazon
<b>Common Name</b> Amazon RSA 2048 M02
<b>Serial Number</b> 01 3D 9A 3F A1 3F 6C 72 93 8F 7A 5C 17 2A E0 88
<b>Version</b> 3
<b>Signature Algorithm</b> SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 )
<b>Parameters</b> None
<b>Not Valid Before</b> Wednesday 3 January 2024 at 00:00:00 Greenwich Mean Time
<b>Not Valid After</b> Saturday 1 February 2025 at 23:59:59 Greenwich Mean Time
<b>Public Key Info</b>
<b>Algorithm</b> RSA Encryption ( 1.2.840.113549.1.1.1 )

# Understanding certificates

- ▶ **Subject:** The entity (individual, server, etc.) to which the certificate belongs
- ▶ **Serial Number:** A unique number identifying the certificate
- ▶ **Validity Period:** The time frame during which the certificate is considered valid
- ▶ **Issuer:** The CA that issued and signed the certificate.

# Understanding certificates



The image shows a certificate chain and its detailed information:

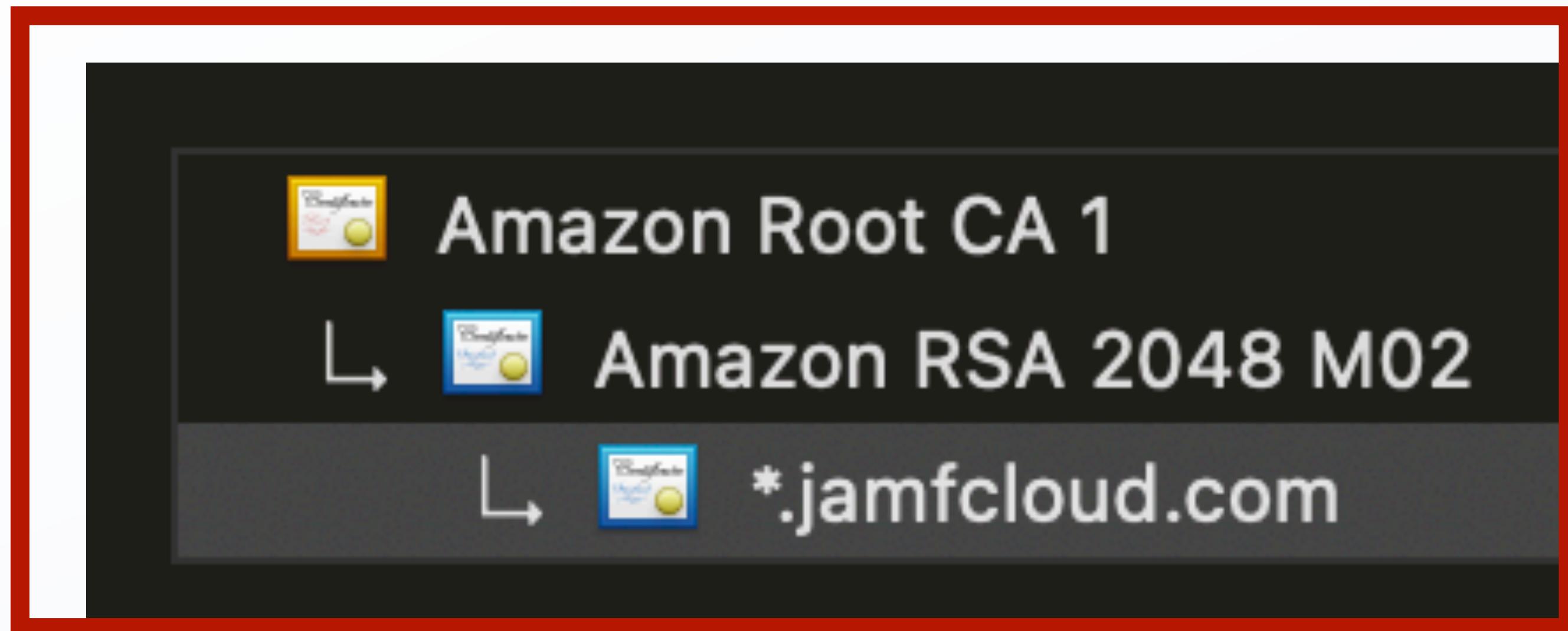
- Root Certificate: Amazon Root CA 1
- Intermediate Certificate: Amazon RSA 2048 M02
- Leaf Certificate: \*.jamfcloud.com

**\*.jamfcloud.com**  
Issued by: Amazon RSA 2048 M02  
Expires: Saturday 1 February 2025 at 23:59:59 Greenwich Mean Time  
 This certificate is valid

**Details**

Subject Name	
Common Name	*.jamfcloud.com
Issuer Name	
Country or Region	US
Organisation	Amazon
Common Name	Amazon RSA 2048 M02

# Understanding certificates



**\*.jamfcloud.com**

Issued by: Amazon RSA 2048 M02

Expires: Saturday 1 February 2025 at 23:59:59

✓ This certificate is valid

# Understanding certificates



► **Root Certificate**



► **Intermediate Certificate**



► **Server or Client Certificate**

# Usage of certificates

- ▶ **802.1x:** Authenticate on a network

# Usage of certificates

- ▶ **802.1x:** Authenticate on a network
- ▶ **VPN:** Authenticate to access a network

# Usage of certificates

- ▶ **802.1x:** Authenticate on a network
- ▶ **VPN:** Authenticate to access a network
- ▶ **S/MIME:** Encrypt email

# Usage of certificates

- ▶ **802.1x:** Authenticate on a network
- ▶ **VPN:** Authenticate to access a network
- ▶ **S/MIME:** Encrypt email
- ▶ **Code Signing:** Sign packages

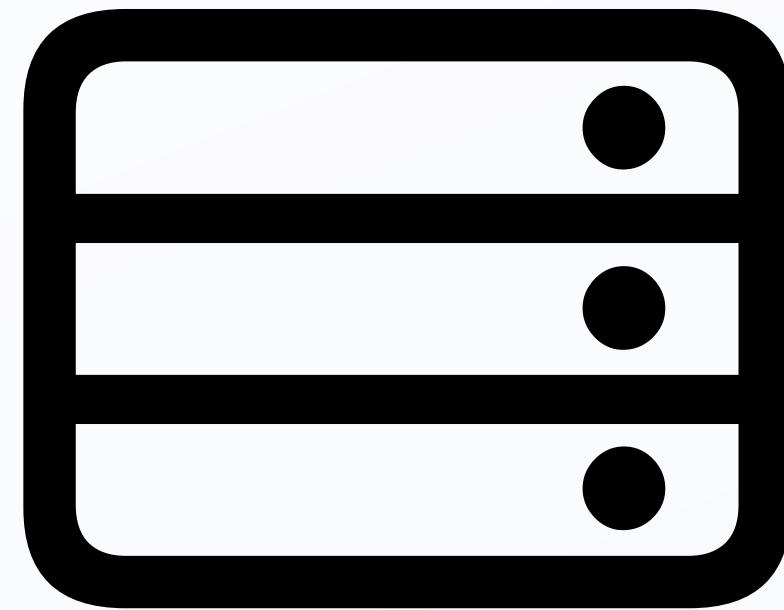
# Usage of certificates



# SCEP

- ▶ Simple Certificate Enrollment Protocol
- ▶ Developed in the late 1990s by Cisco Systems, publicly available around 2000
- ▶ The initial use cases was securing network devices such as routers, switches, and firewalls
- ▶ HTTP

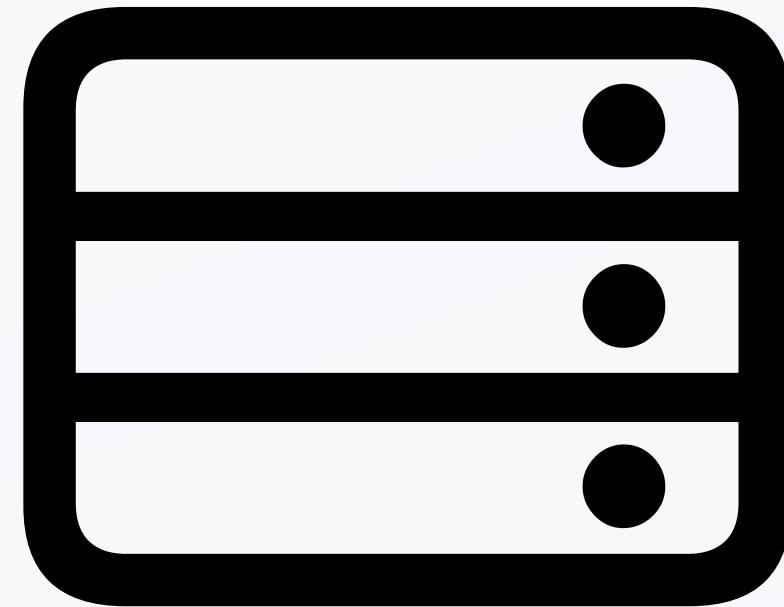
# SCEP



SCEP

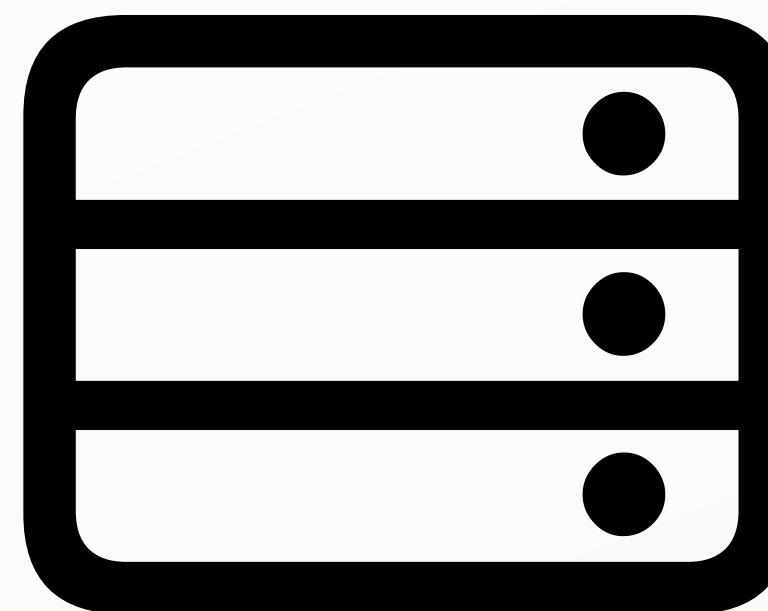


Device



Jamf Pro

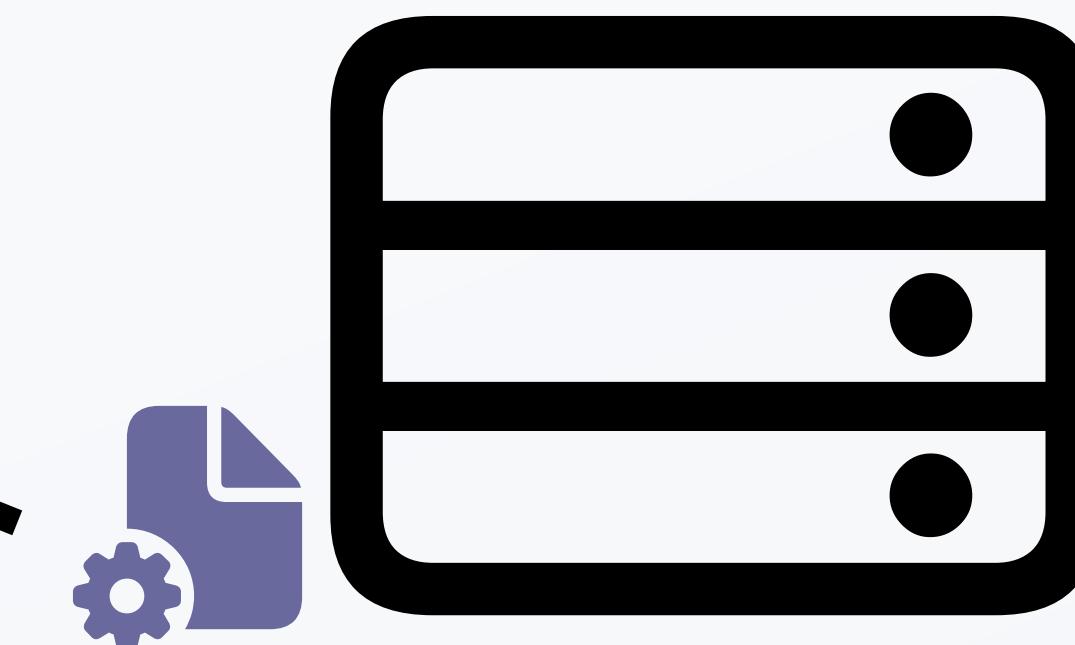
# SCEP



SCEP

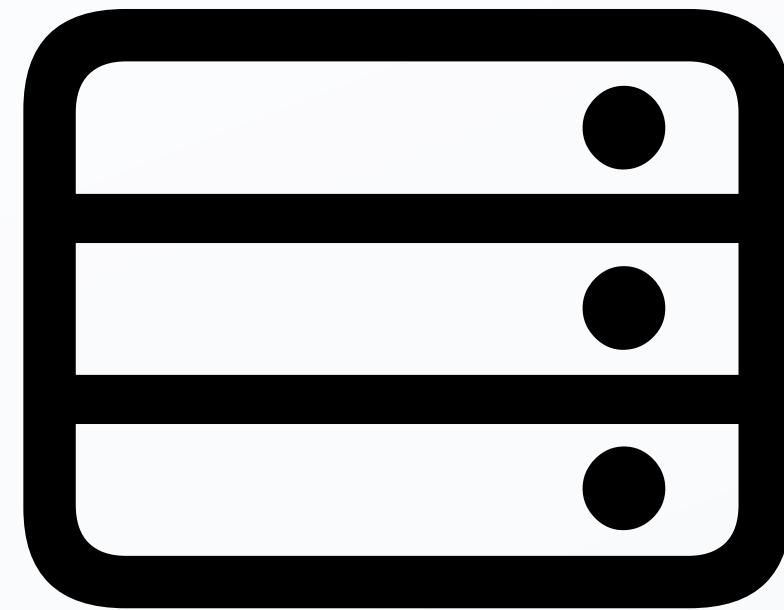


Device

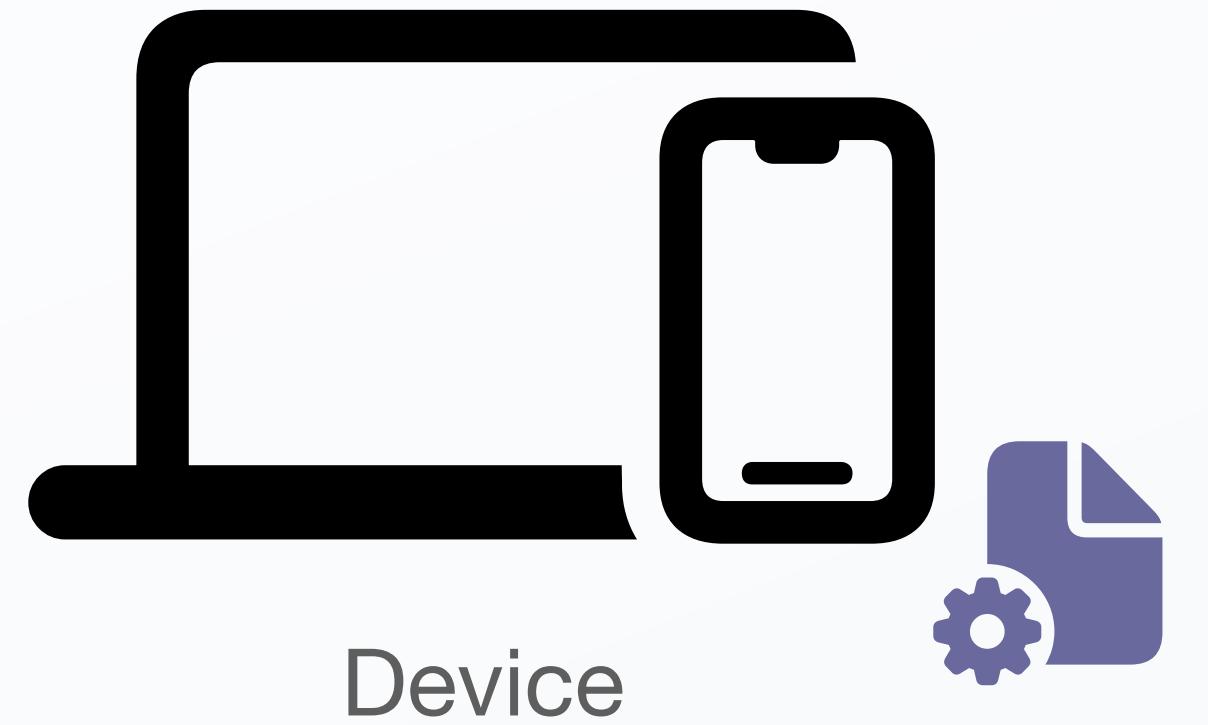


Jamf Pro

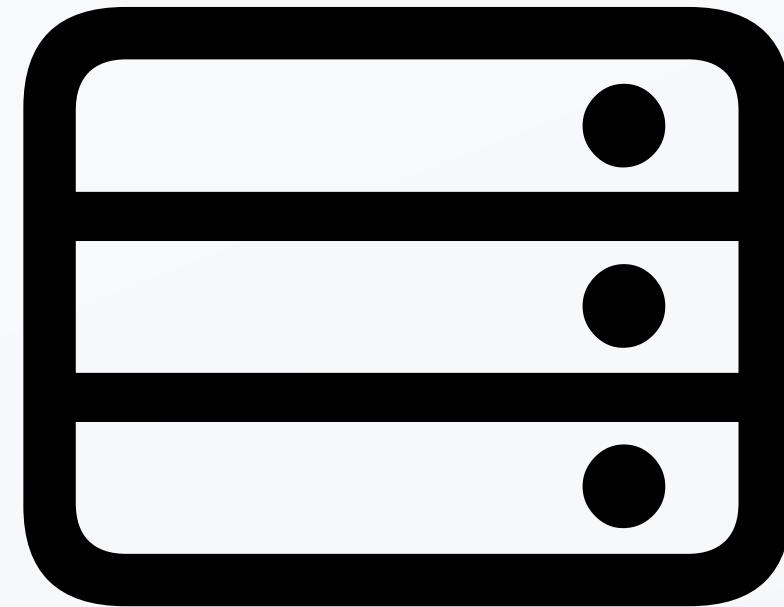
# SCEP



SCEP

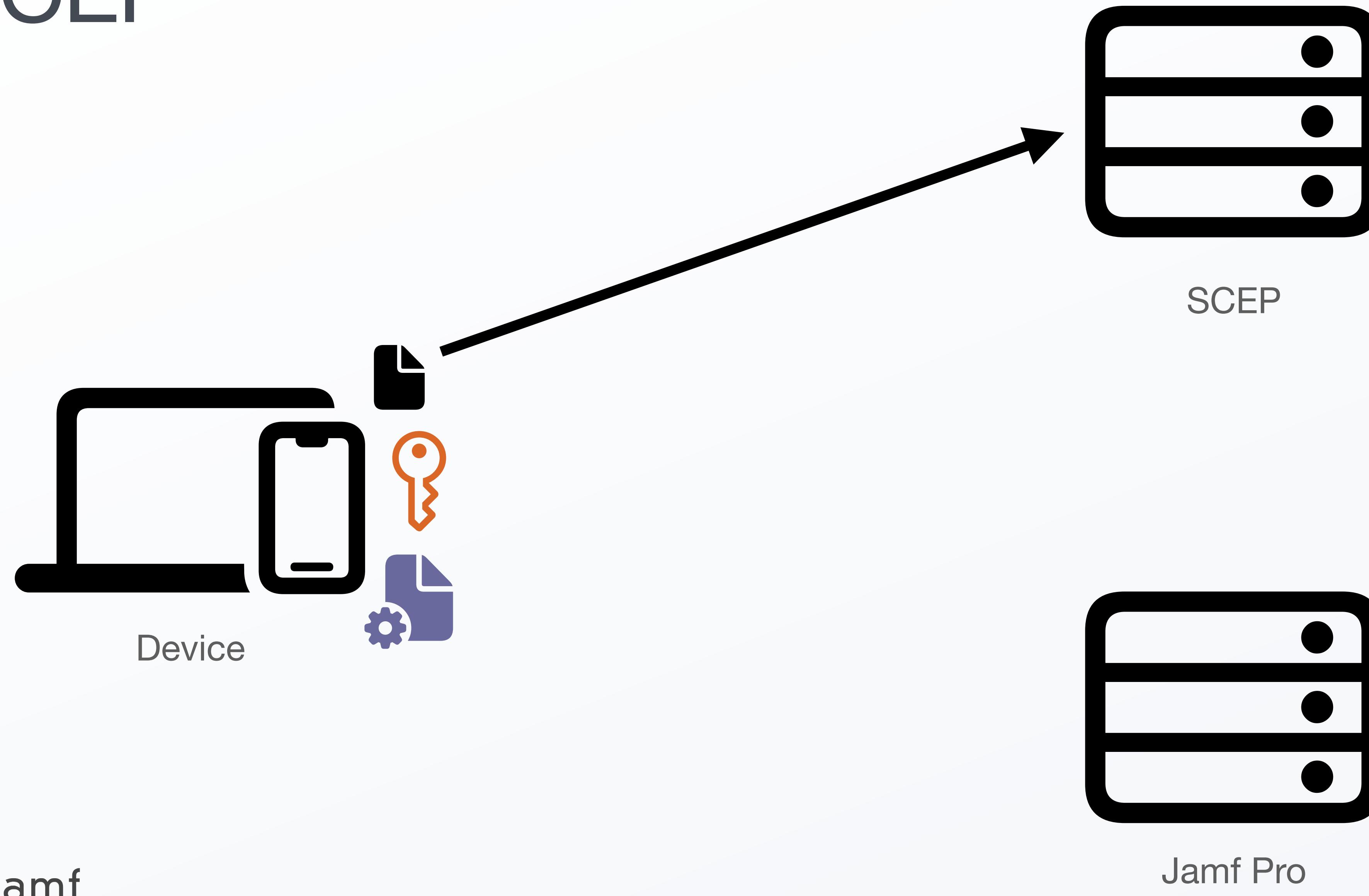


Device

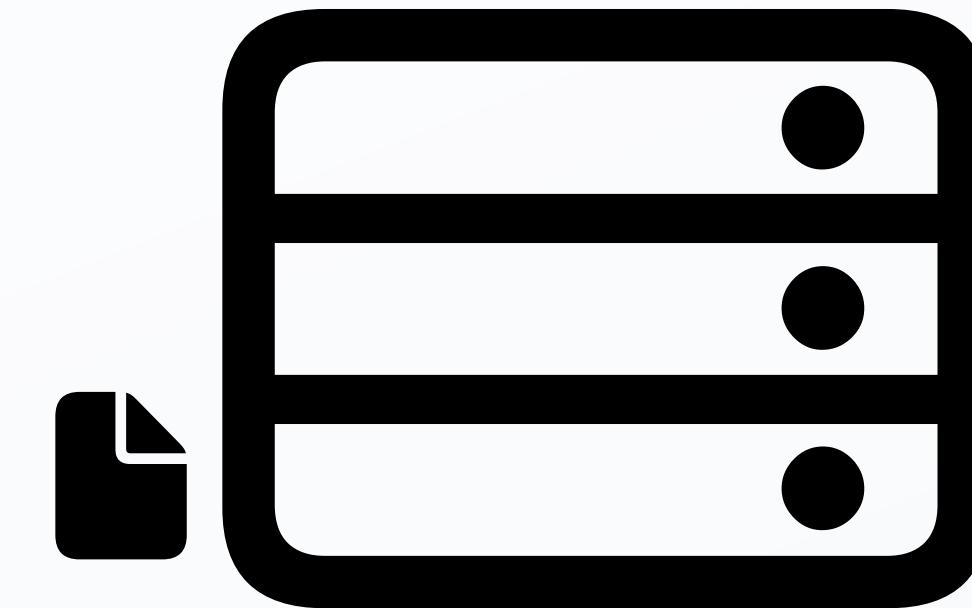


Jamf Pro

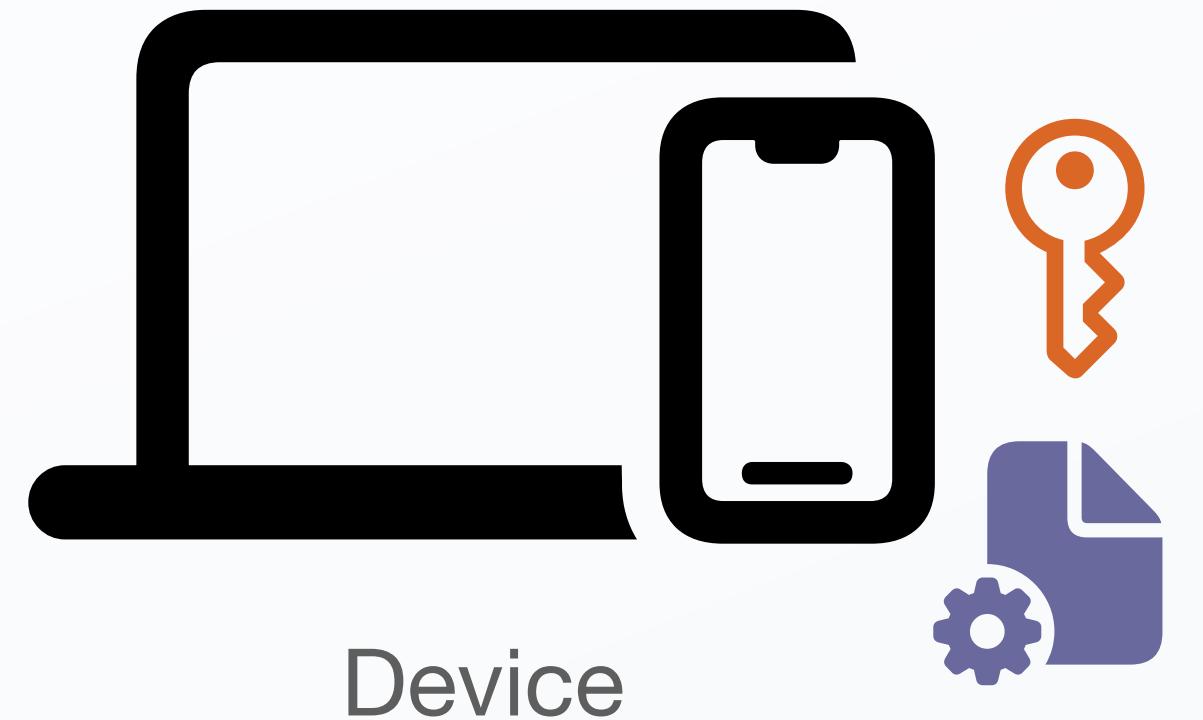
# SCEP



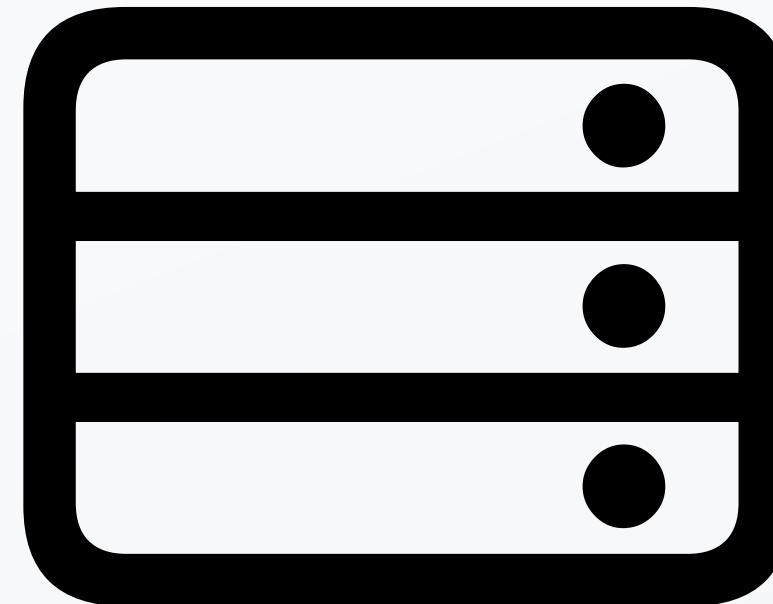
# SCEP



SCEP

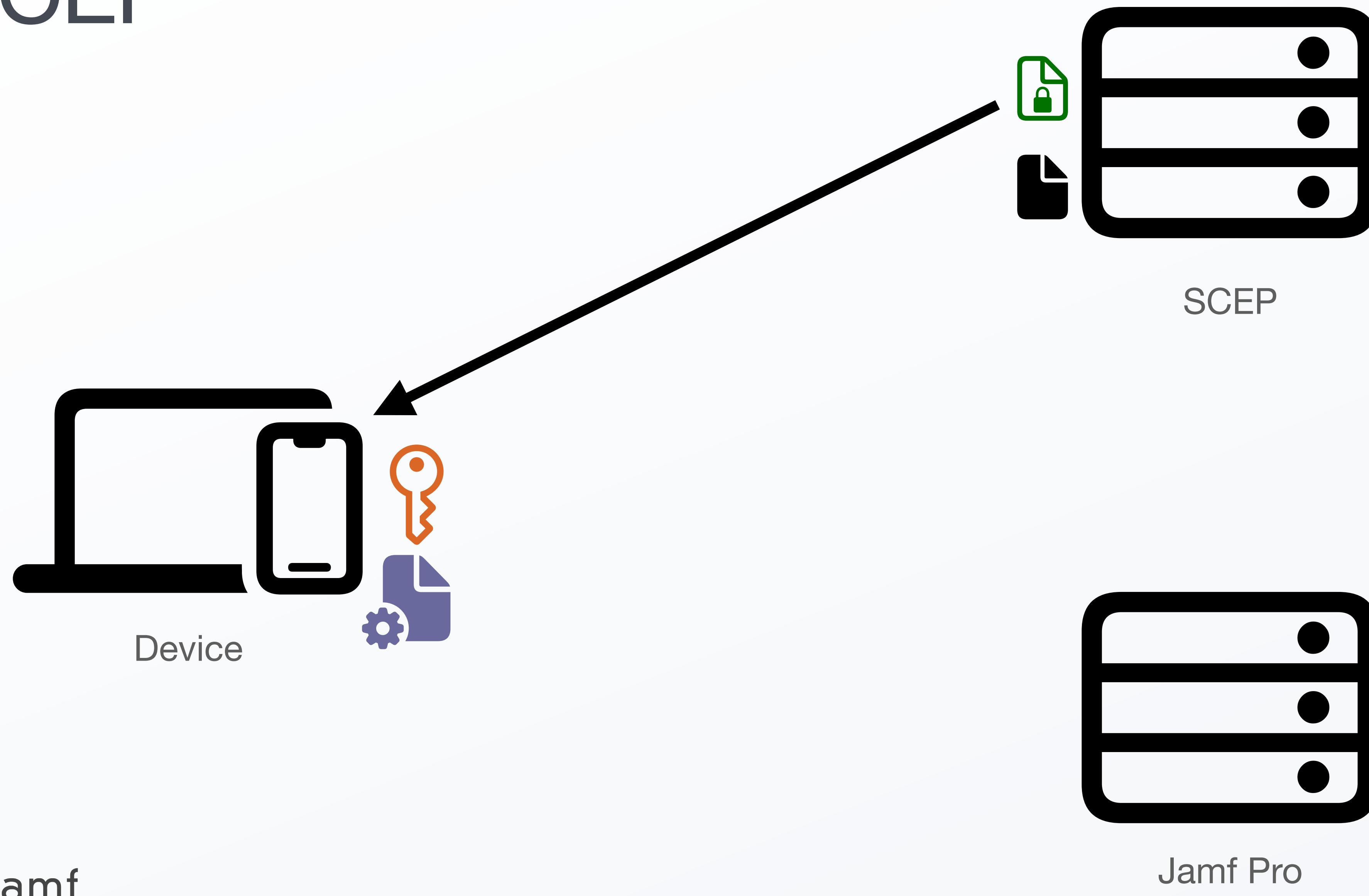


Device

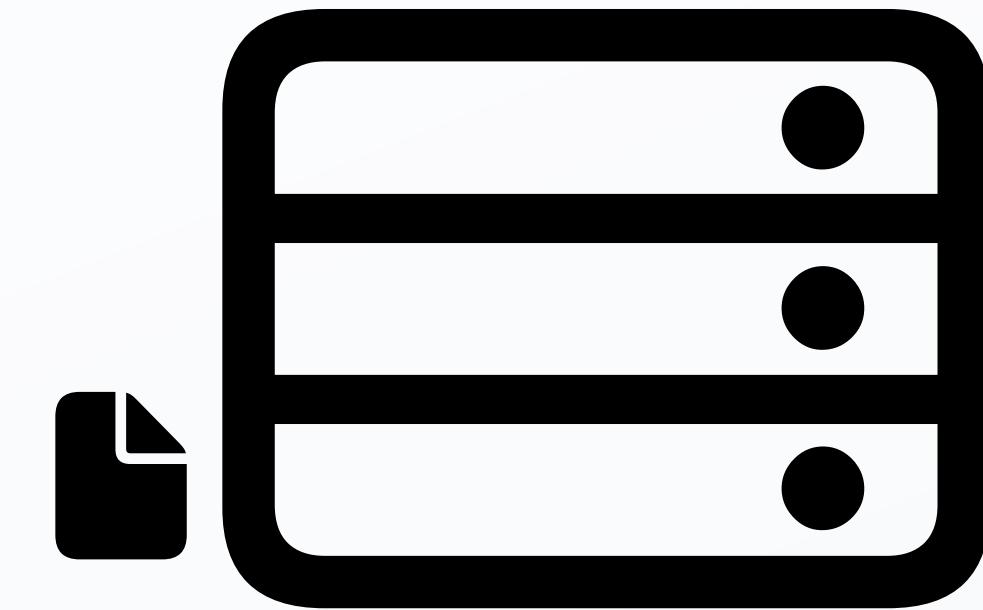
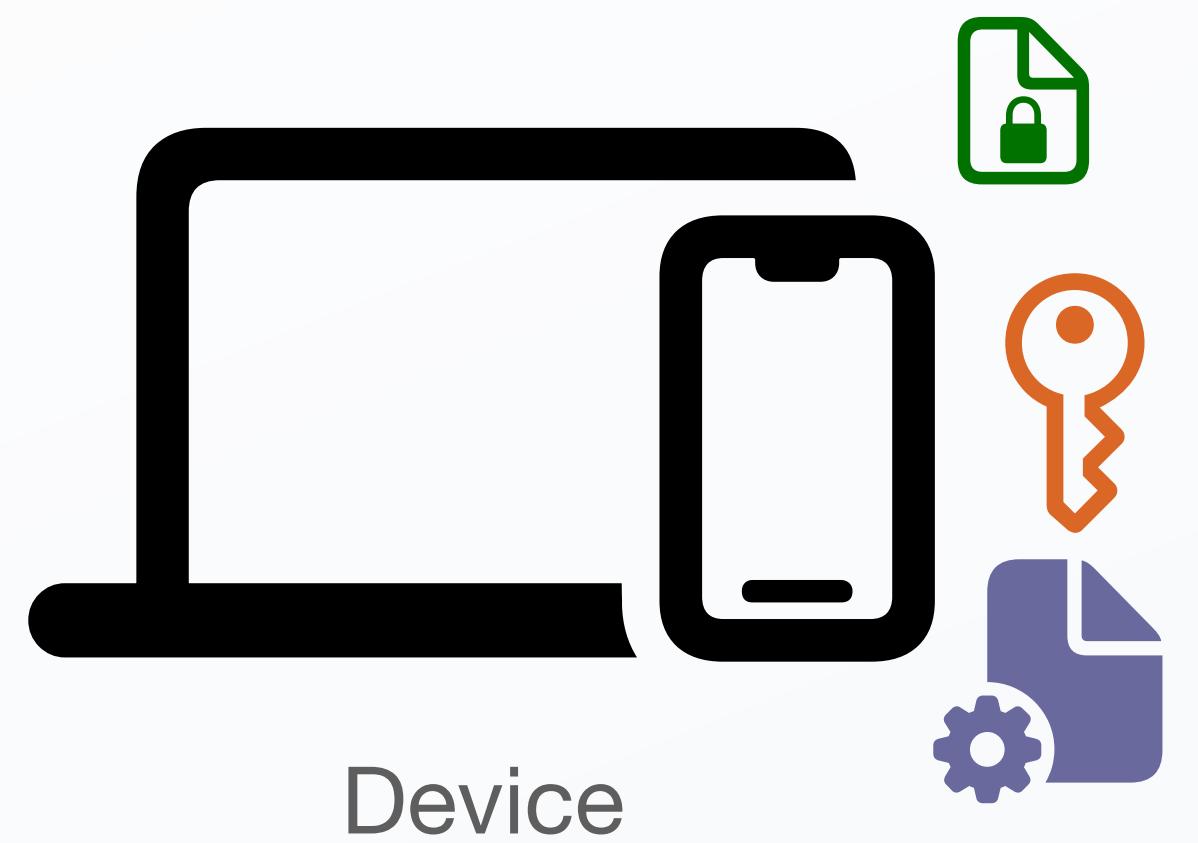


Jamf Pro

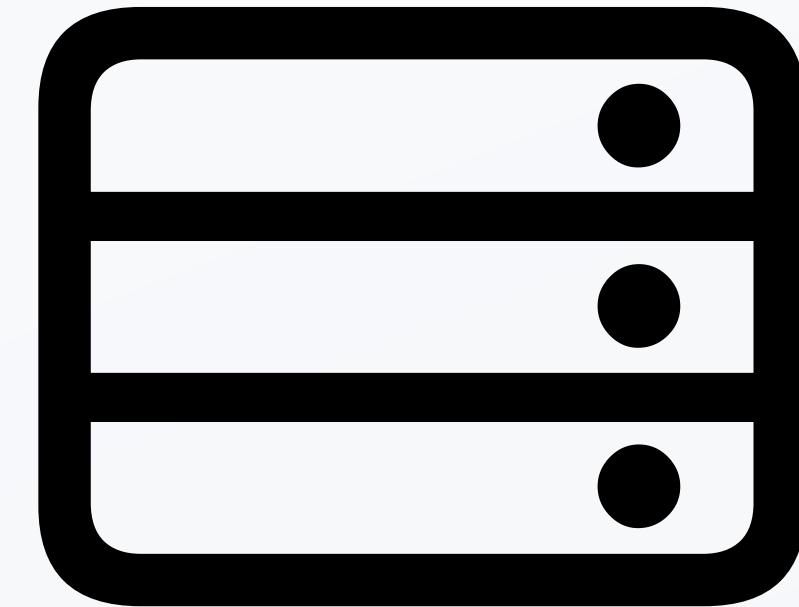
# SCEP



# SCEP



SCEP



Jamf Pro

# ACME

- ▶ Automatic Certificate Management Environment
- ▶ Developed by Internet Security Research Group (ISRG)
- ▶ 2015 adopted by Let's Encrypt
- ▶ WWDC 2022 - Managed Device Attestation

# ACME



Published Nov 7, 2017 • 4 min read

## Device provisioning: Identity attestation with TPM

By [Nicole Berdy](#), Principal Program Manager, Office of the CTO

**SHARE**

**CONTENT TYPE**

Best practices

**AUDIENCE**

more ▾

Folks using the [IoT Hub Device Provisioning Service](#) to securely provision their devices are taking the opportunity to start using hardware security modules (HSM) to store the keys on their devices. Hardware security modules protect cryptographic keys and operations. HSMs provide high levels of protection against key compromise by device software and firmware bugs, and usually provide good protection against hardware attacks. Hardware-based security can reduce the risk of device cloning, can improve supply-chain security, and can bootstrap secure and reliable device enrollment using the Device Provisioning Service. Some of you might be new to using HSMs and are wondering exactly how the Device Provisioning Service validates a device's identity, especially when using TPMs, and why it's so secure. This post describes the identity attestation process when using a TPM.

TPM stands for Trusted Platform Module and is a type of HSM. This blog post assumes you're using a discrete, firmware, or integrated TPM. Software

# ACME



# Google



Used by 168

Code of conduct Apache-2.0 license Security

## Go-Attestation

[reference](#)

Go-Attestation abstracts remote attestation operations across a variety of platforms and TPMs, enabling remote validation of machine identity and state. This project attempts to provide high level primitives for both client and server logic.

Talks on this project:

- "Making Device Identity Trustworthy" - Open Source Summit Europe - October 2019 - ([Slides](#))
- "Using TPMs to Cryptographically Verify Devices at Scale" - Open Source Summit North America - September 2019 - ([Video](#) 39min)
- "Making Remote Attestation Useful on Linux" - Linux Security Summit - September 2019 - ([Video](#) 26min)

### Status

Go-Attestation is under active development. Expect API changes at any time.

Please note that this is not an official Google product.

TPM 1.2 support is best effort, meaning we will accept fixes for TPM 1.2, but testing is not covered by CI.

### Installation

The go-attestation package is installable using go get: `go get github.com/google/go-attestation/attest`

### TPM1.2

Contributors 36

+ 22 contributors

Languages

Go 98.0% Shell 2.0%

# ACME



# Google



# SAMSUNG

**SAMSUNG Knox Documentation** Admin Developer Partner  Search documentation

Samsung Knox developer guides ▾

**Knox Attestation**

- Introduction
- Enhanced Attestation (v3)**
- Tutorial ▾
- API reference ▾
- Release notes

Home / Knox Attestation /

## Enhanced Attestation (v3)

Samsung Knox Enhanced Attestation is a feature that verifies a Samsung device's data integrity by checking that the device isn't rooted or running unofficial firmware.

**Warning**

Knox 3.4 introduced the latest version of Attestation (v3) running on flagship devices from the Note 10 onwards. Enhanced Attestation uses the [EnhancedAttestationPolicy](#) class and [v3 REST API](#). For information about the previous version of Attestation, see [Attestation \(v2\)](#).

### About Enhanced Attestation

#### Samsung Attestation Key

Enhanced Attestation uses the Samsung Attestation Key (SAK) to prove:

- The key is protected by a secure hardware.
- The device is manufactured by Samsung.
- The device ID isn't compromised.

When verifying devices as Samsung devices, it's important to note that certificate change alone isn't enough to prove that a device is a Samsung device since malicious attackers can send a certificate chain generated by

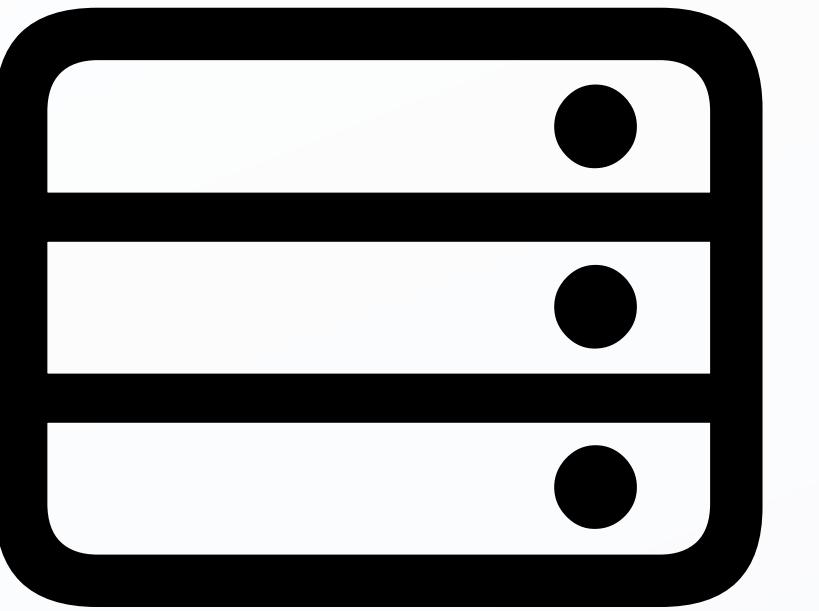
**On this page**

- [About Enhanced Attestation](#)
- [Samsung Attestation Key](#)
- [Enhanced Attestation process](#)
- [Signature](#)
- [Certificate and Verification](#)
- [Secure communication](#)
- [How Attestation works](#)

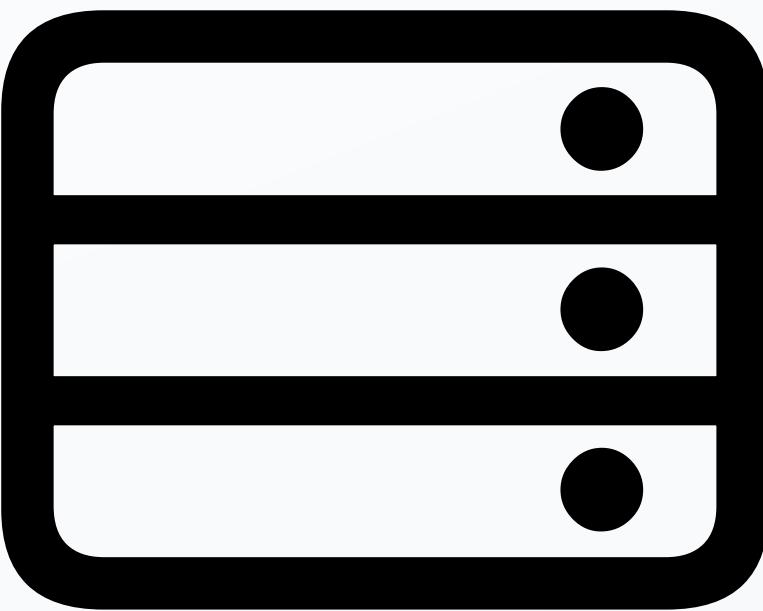
# ACME

<https://datatracker.ietf.org/doc/html/draft-acme-device-attest>

# ACME



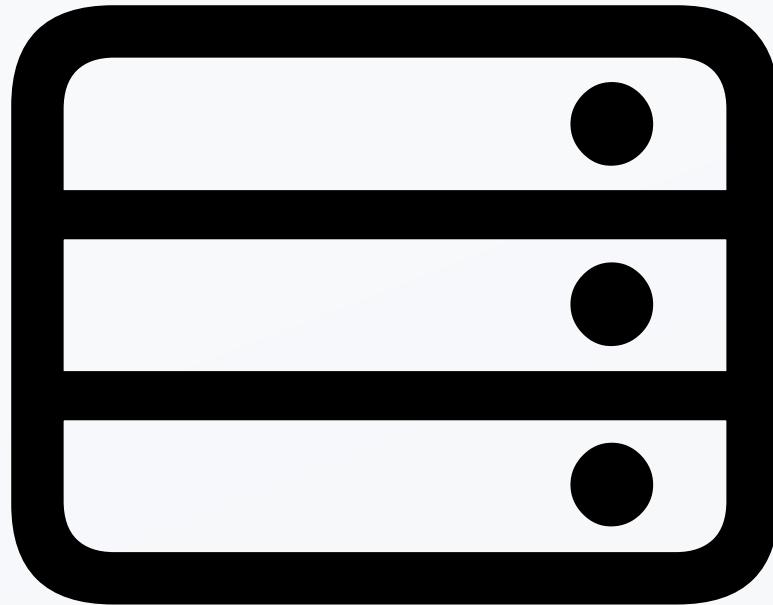
Attestation  
Server



ACME Server

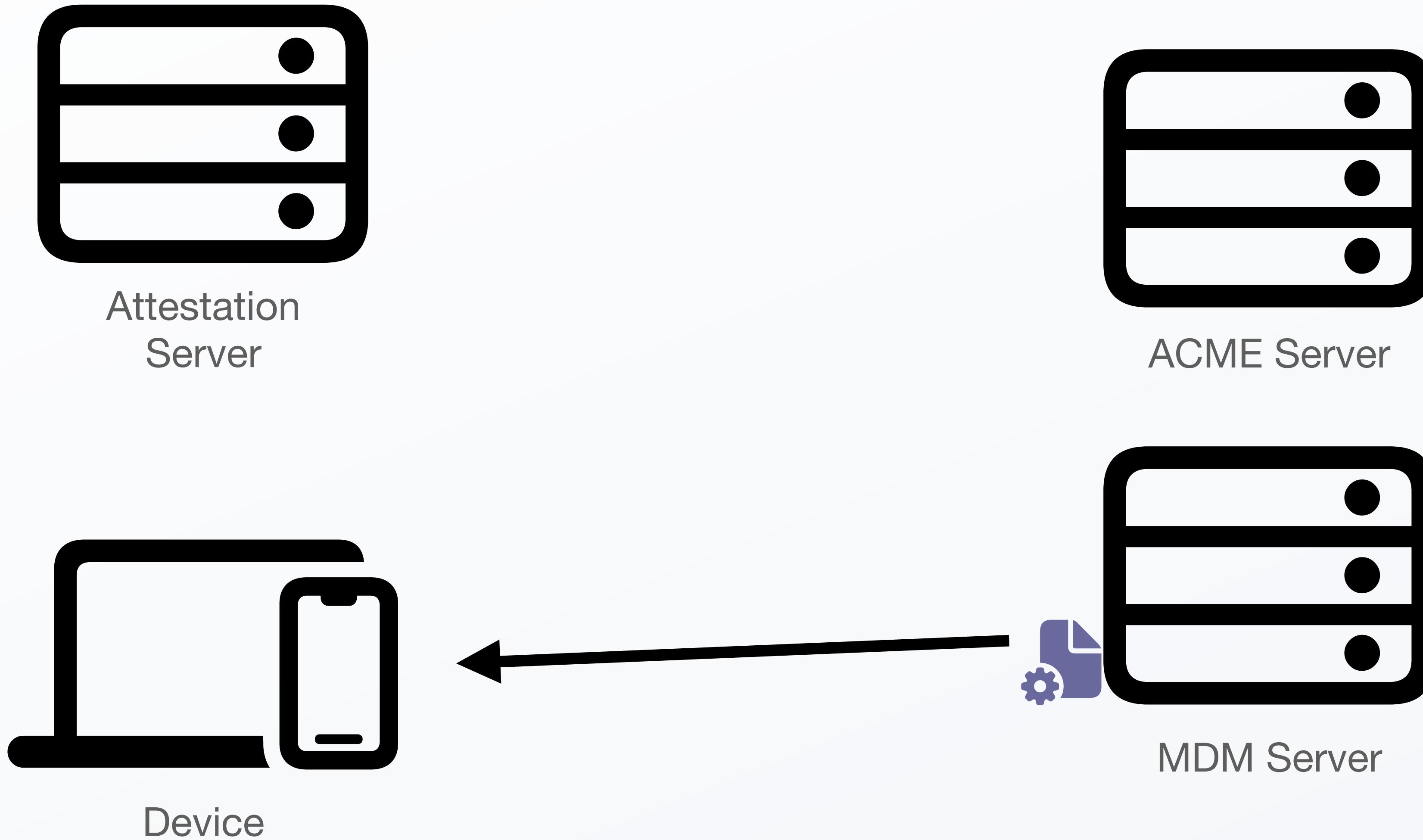


Device

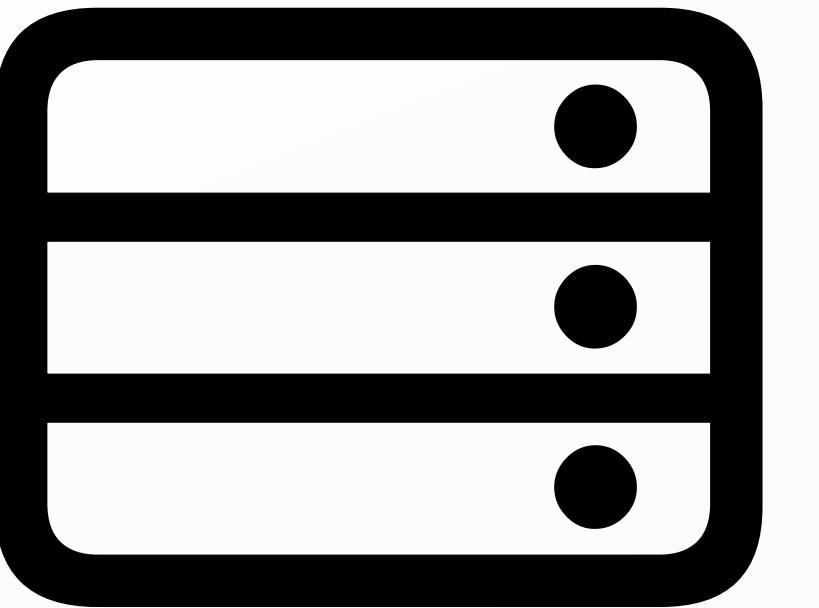


MDM Server

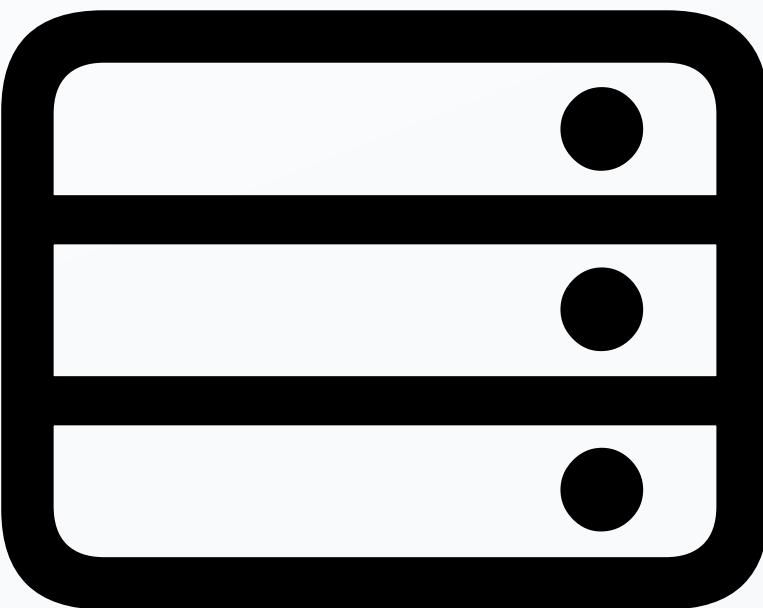
# ACME



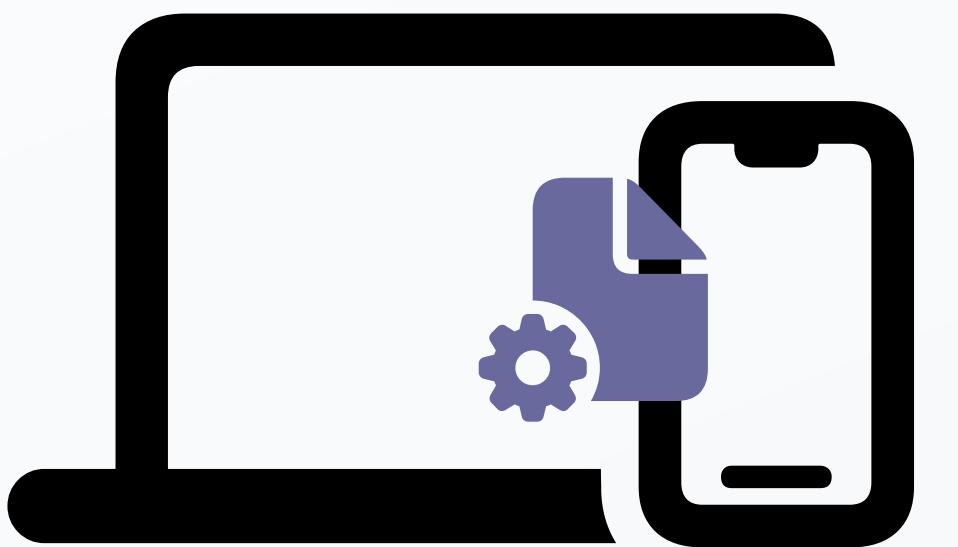
# ACME



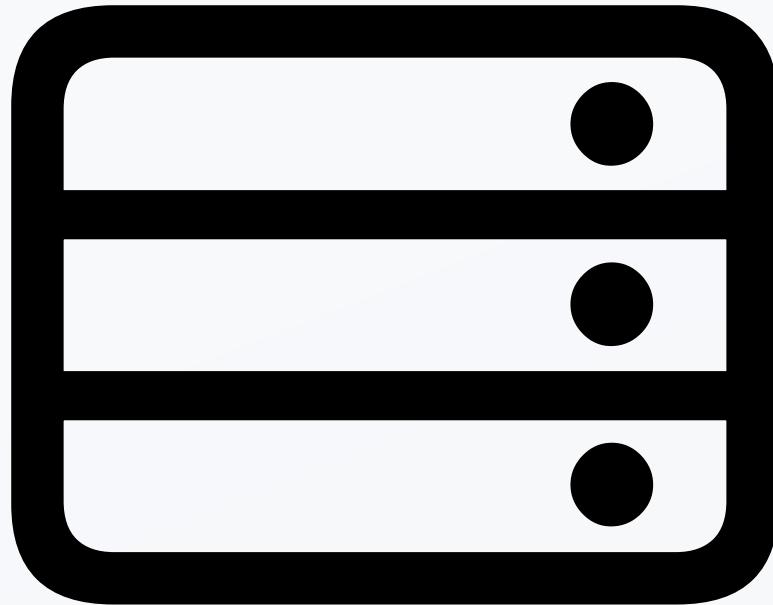
Attestation  
Server



ACME Server



Device



MDM Server

# ACME

Payload keys

PayloadType com.apple.security.acme

# Usage of certificates

## Payload keys

```
PayloadType com.apple.security.acme
```

## Key properties

```
KeyType  
KeySize  
HardwareBound
```

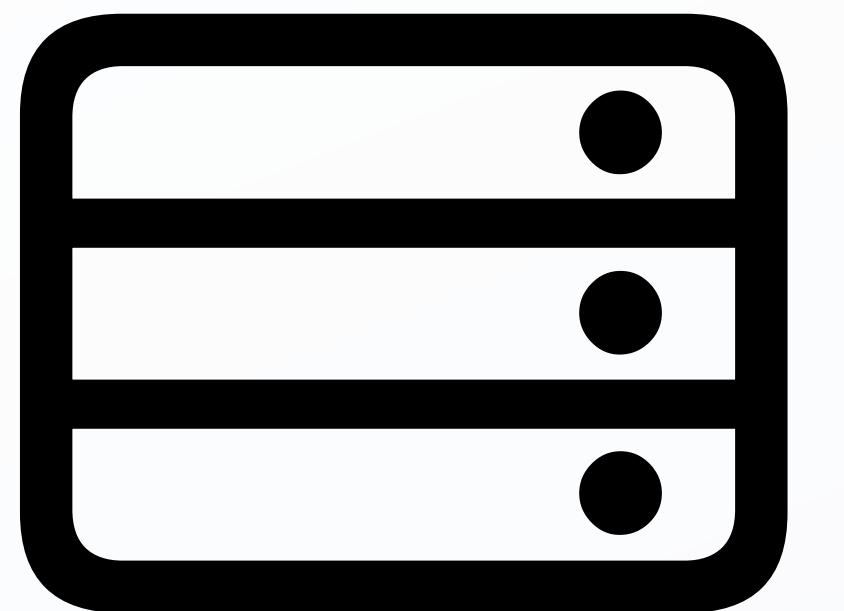
## Certificate request

```
KeyUsage  
ExtendedKeyUsage  
Subject  
SubjectAltName
```

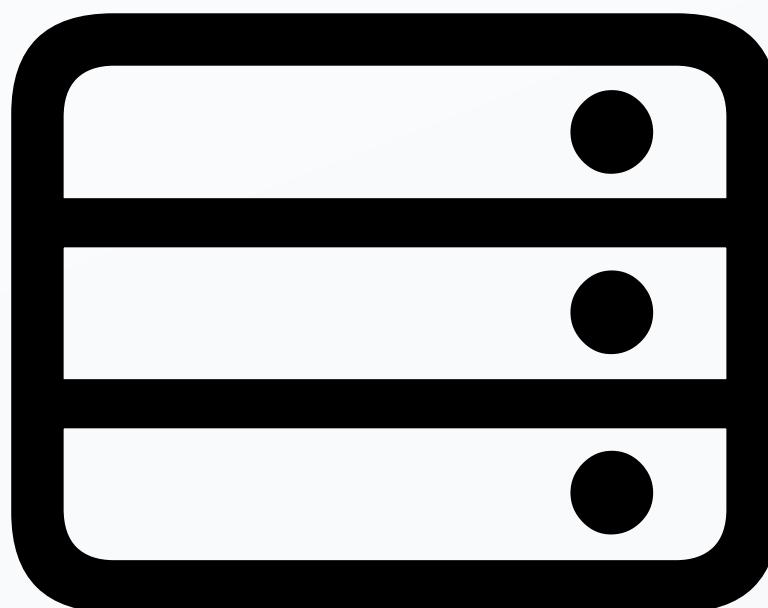
## ACME request

```
DirectoryURL  
ClientIdentifier  
Attest
```

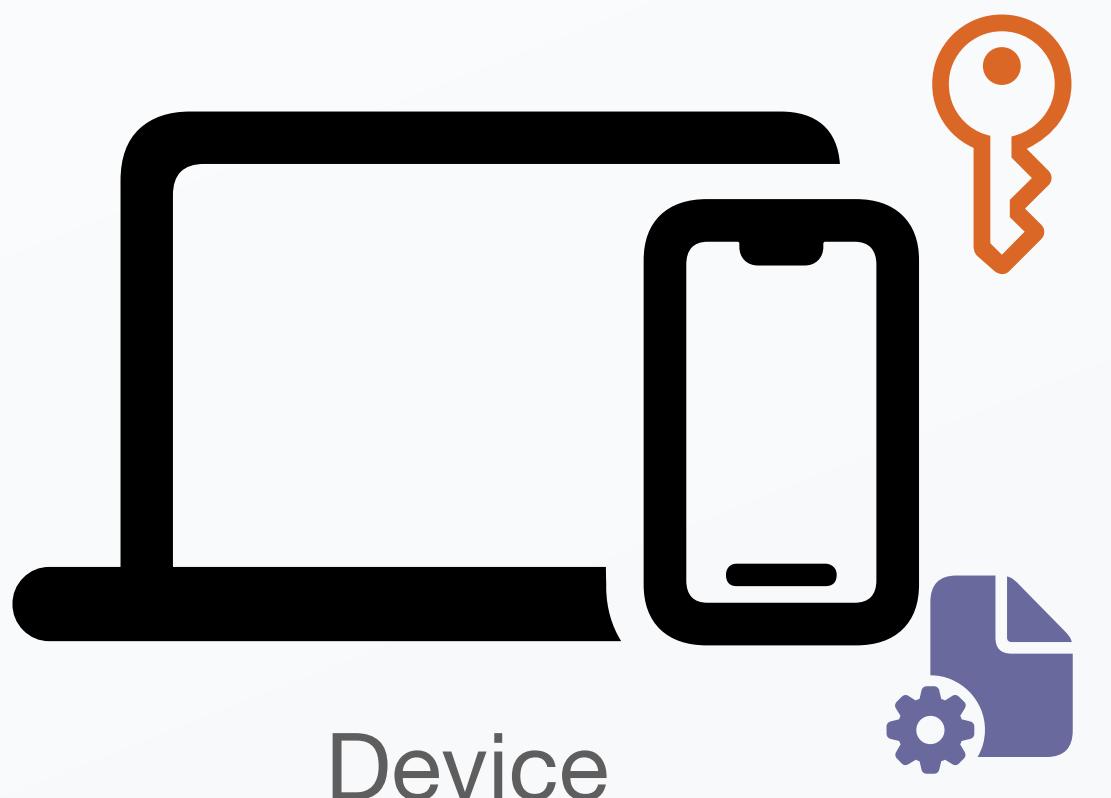
# ACME



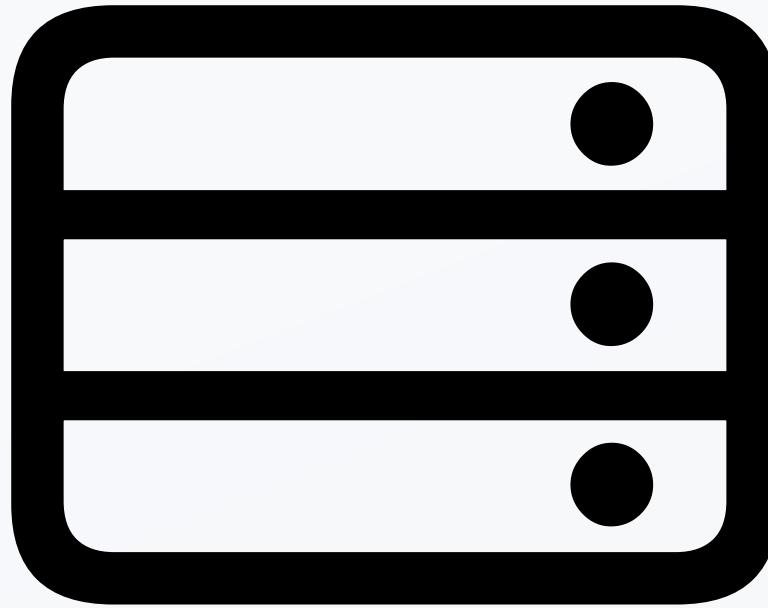
Attestation  
Server



ACME Server



Device

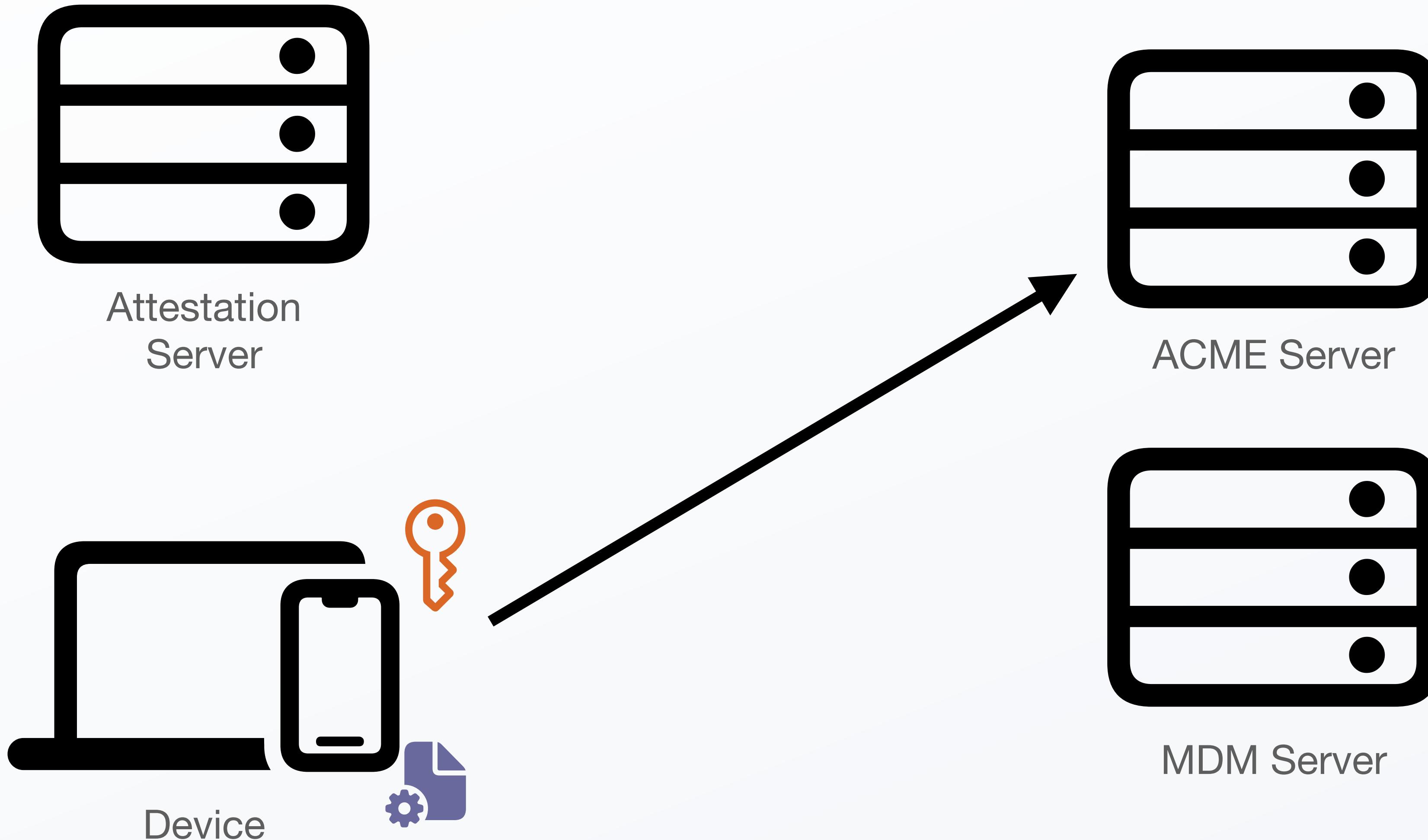


MDM Server

# ACME

- ▶ Attestation required hardware-bound key
- ▶ ACME payload support **RSA** and **ECSECPrimeRandom**
- ▶ Hardware-bound keys requires **ECSECPrimeRandom**

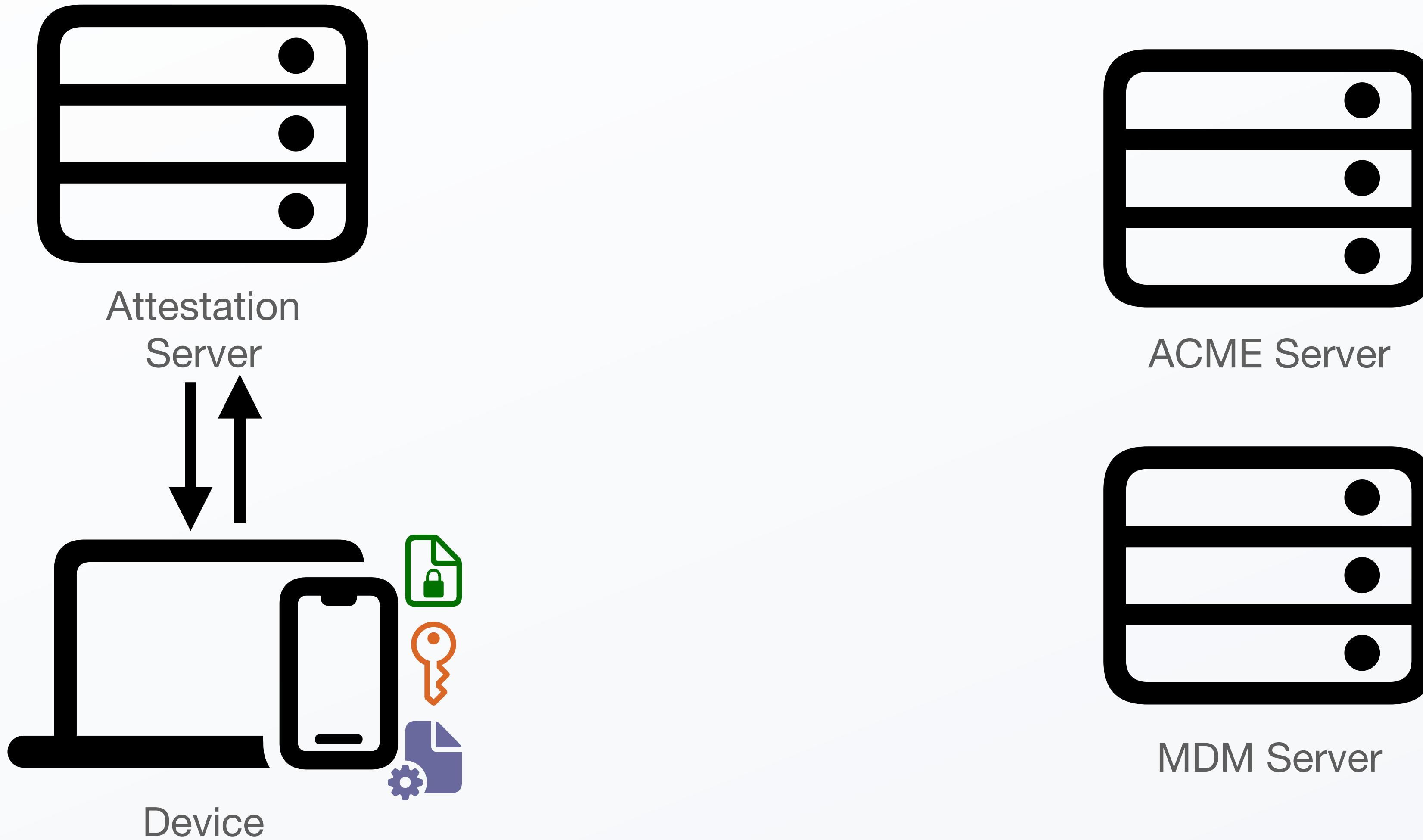
# ACME



# ACME

- ▶ Resolve DirectoryURL
- ▶ Account and order creation
- ▶ device-attest-01 validation type
- ▶ Generate nonce token

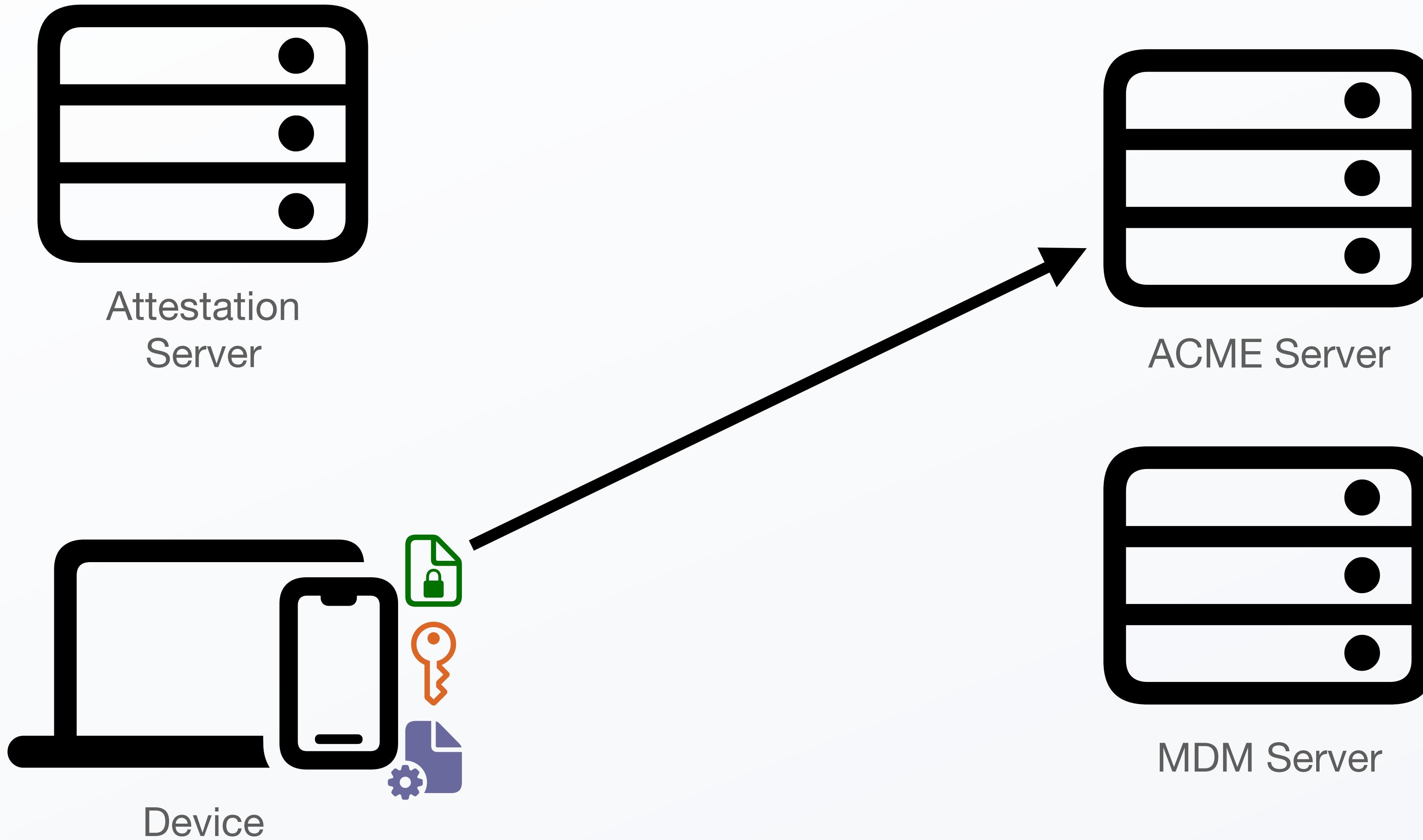
# ACME



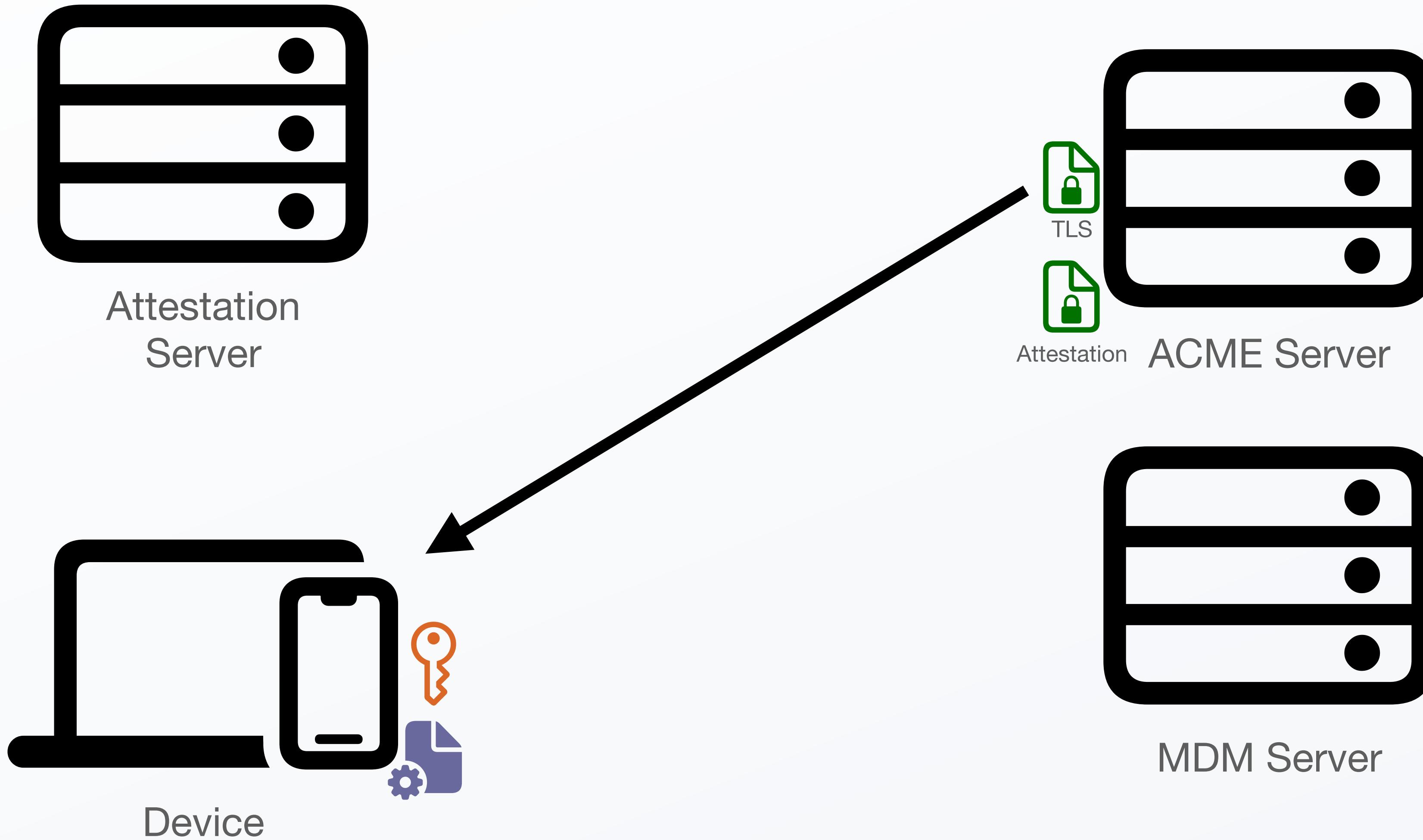
# ACME

- ▶ Nonce hashed (SHA-256) before embedding in attestation
- ▶ OIDs - Device-Identifying are omitted for User Enrollments
- ▶ Nonce from ACME server, not MDM
- ▶ Private key is the one the device generated
- ▶ Certificate can be used only for Attestation

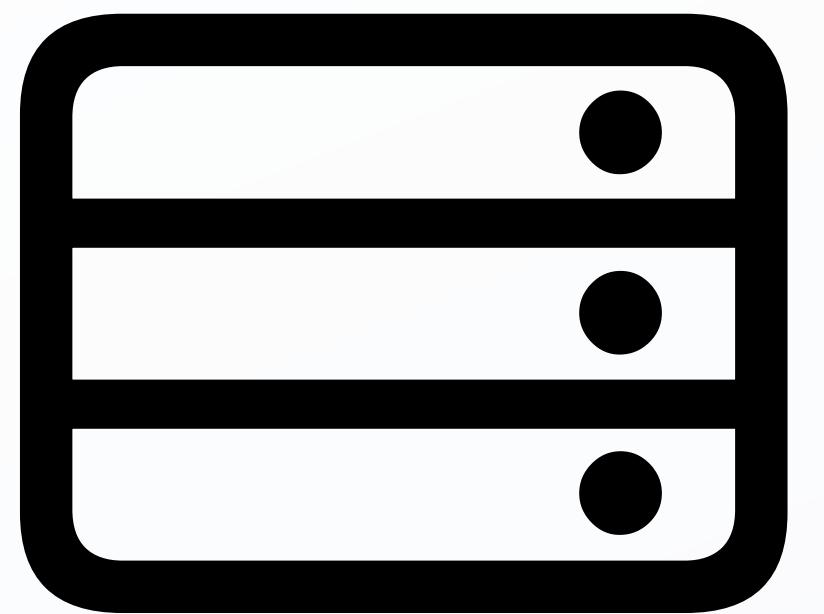
# ACME



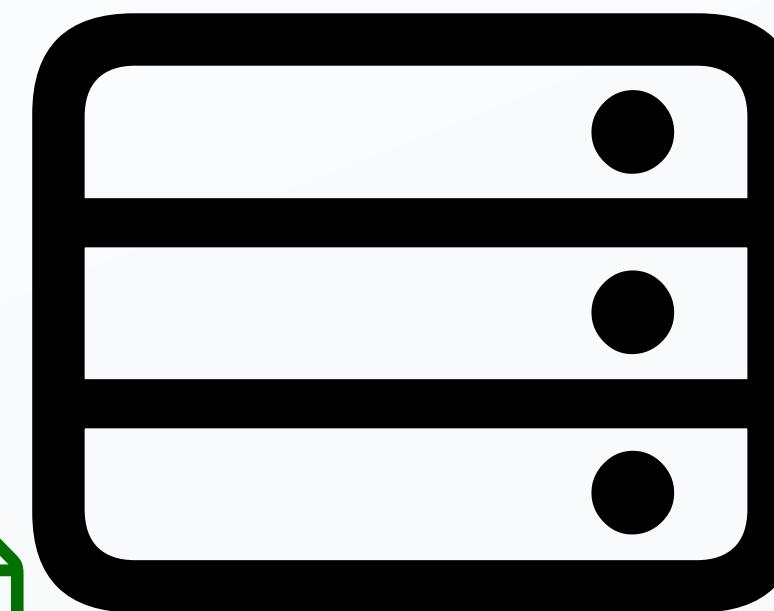
# ACME



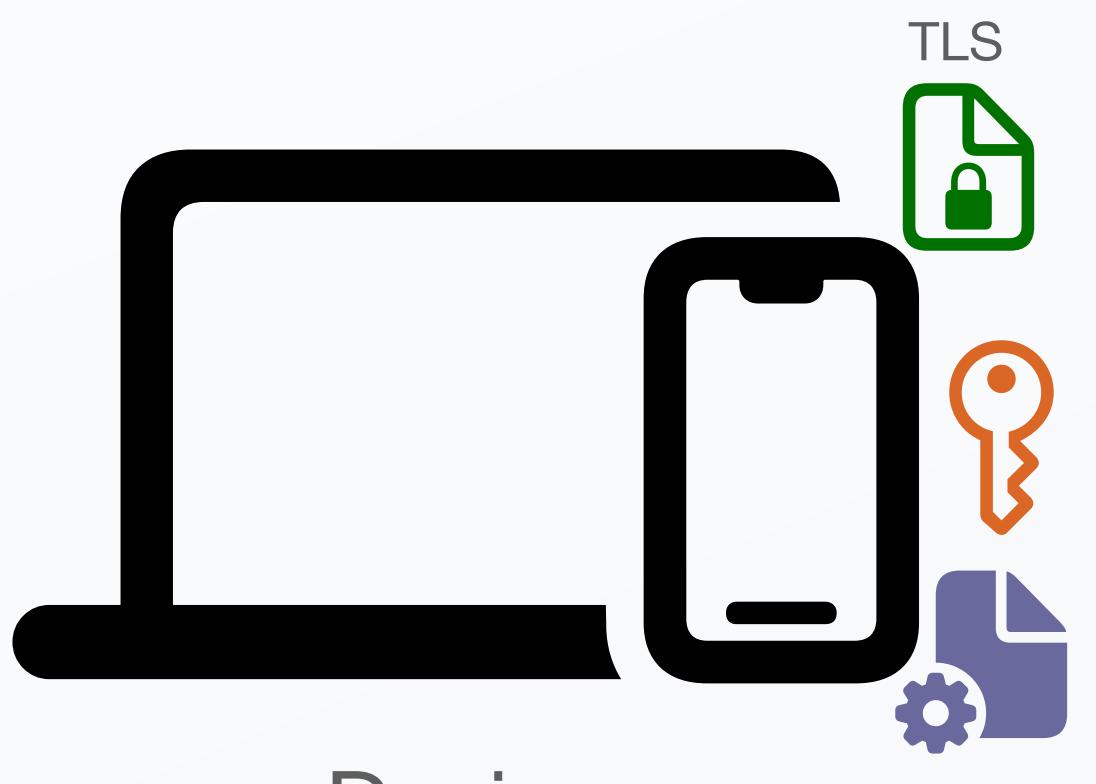
# ACME



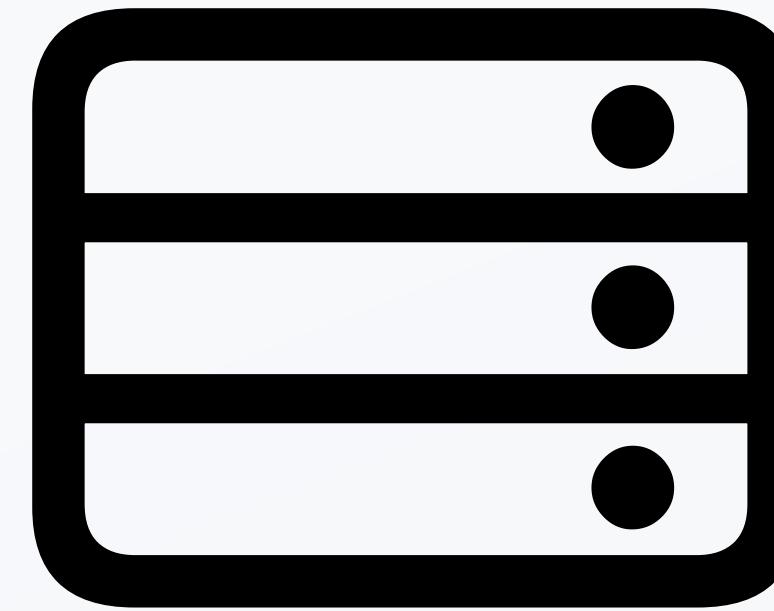
Attestation  
Server



Attestation ACME Server



Device



MDM Server

# ACME with Jamf Pro

The screenshot shows the Jamf Pro web interface. The left sidebar has a dark theme with white icons and text. The 'Computers' section is expanded, showing 'Inventory', 'Search Inventory', 'Search Volume Content', and 'Licensed Software'. Under 'Content Management', there are 'Policies', 'Configuration Profiles' (which is selected and highlighted in blue), 'Software Updates', and 'Restricted Software'. The main content area is titled 'New macOS Configuration Profile' under 'Computers : Configuration Profiles'. It shows two tabs: 'Options' (which is selected) and 'Scope'. A search bar says 'Search...'. Below it, there's a list of settings: 'General' (with a gear icon), 'Accessibility' (with a person icon, labeled 'Not configured'), 'ACME Certificate' (with a gear icon, labeled 'Not configured'), and 'AD Certificate' (with a shield icon, labeled 'Not configured'). To the right of the list are two buttons: 'Remove all' and '+ Add'. At the top right of the interface, there are status indicators: 'Full Jamf Pro' with a dropdown arrow, a bell icon with a red circle containing '13', and a user profile icon.

Computers : Configuration Profiles

← New macOS Configuration Profile

Options Scope

Search...

ACME Certificate

The payload you use to configure Automated Certificate Management Environment (ACME) Certificate settings.

General

Accessibility Not configured

ACME Certificate Not configured

AD Certificate Not configured

Remove all + Add

# ACME with Jamf Pro

The screenshot shows the Jamf Pro web interface. On the left, a sidebar navigation bar includes icons for Computers, Inventory, Search Inventory, Search Volume Content, Licensed Software, Content Management, Policies, Configuration Profiles (which is selected and highlighted in blue), Software Updates, Restricted Software, Mac Apps, Patch Management, eBooks, Groups, and Smart Computer Groups. The main content area displays a list of Configuration Profiles for the computer 'ACME'. The 'ACME Certificate' profile is currently selected. A modal window titled 'ACME Certificate' provides detailed configuration settings for this profile, including the 'ACME directory URL' set to 'https://acme.jamf.net/acme/acme/directory/' and the 'Client identifier' set to '\$SERIALNUMBER'. The modal also includes sections for 'Key Size' and 'Key Type', both of which are currently set to 'Required'. At the bottom right of the modal are 'Cancel' and 'Save' buttons. The top right corner of the screen shows a notification badge with the number '13'.

# ACME with Jamf Pro

## ACME Certificate

Configure ACME Certificate settings.

### ACME directory URL

The directory URL of the ACME server. The URL must use the https scheme.

`https://acme.jamf.net/acme/acme/directory/`

Required

### Client identifier

A unique string identifying a specific device. The server may use this as a nonce to prevent issuing multiple certificates. This identifier also indicates to the ACME server that the device has access to a valid client identifier issued by the enterprise infrastructure. This can help the ACME server determine whether to trust the device. Though this is a relatively weak indication because of the risk that an attacker can intercept the client identifier.

`$SERIALNUMBER`

Required



The screenshot shows the Jamf Pro web interface with the 'ACME Certificate' configuration page open. The left sidebar lists various management categories like Search Inventory, Content Management, Policies, and Smart Computer Groups. The right pane is titled 'ACME Certificate' and contains two main sections: 'ACME directory URL' and 'Client identifier'. The 'ACME directory URL' field is populated with 'https://acme.jamf.net/acme/acme/directory/'. The 'Client identifier' field is populated with '\$SERIALNUMBER'. Both fields have a 'Required' status indicator. At the bottom of the configuration pane, there are 'Save' and 'Cancel' buttons.

# ACME with Jamf Pro

## Key Type

The type of key pair to generate.

- **RSA** : Specifies an RSA key pair. RSA key pairs must have a **Key Size** in the range **[1024..4096]** inclusive and a multiple of 8, and **Hardware Bound** must be **false**.
- **ECSECPrimeRandom** : Specifies a key pair on the P-192, P-256, P-384 or P-521 curves as defined in FIPS Pub 186-4. **Key Size** defines the particular curve, which must be **192**, **256**, **384** or **521**. **Hardware bound** keys only support values of **256** and **384**.

Note that the key size is **521**, not **512**, even though the other key sizes are multiples of 64.

ECSECPrimeRandom



## Hardware Bound

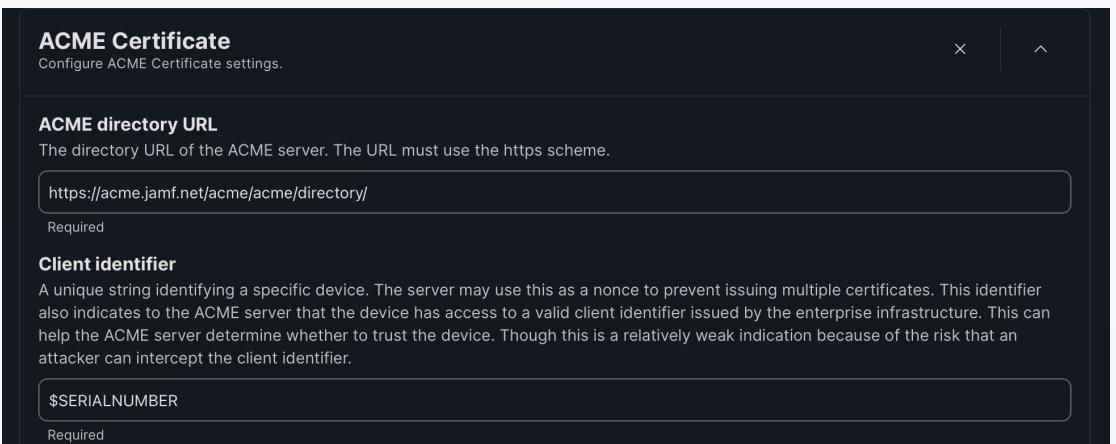
If **false**, the private key isn't bound to the device.

If **true**, the private key is bound to the device.

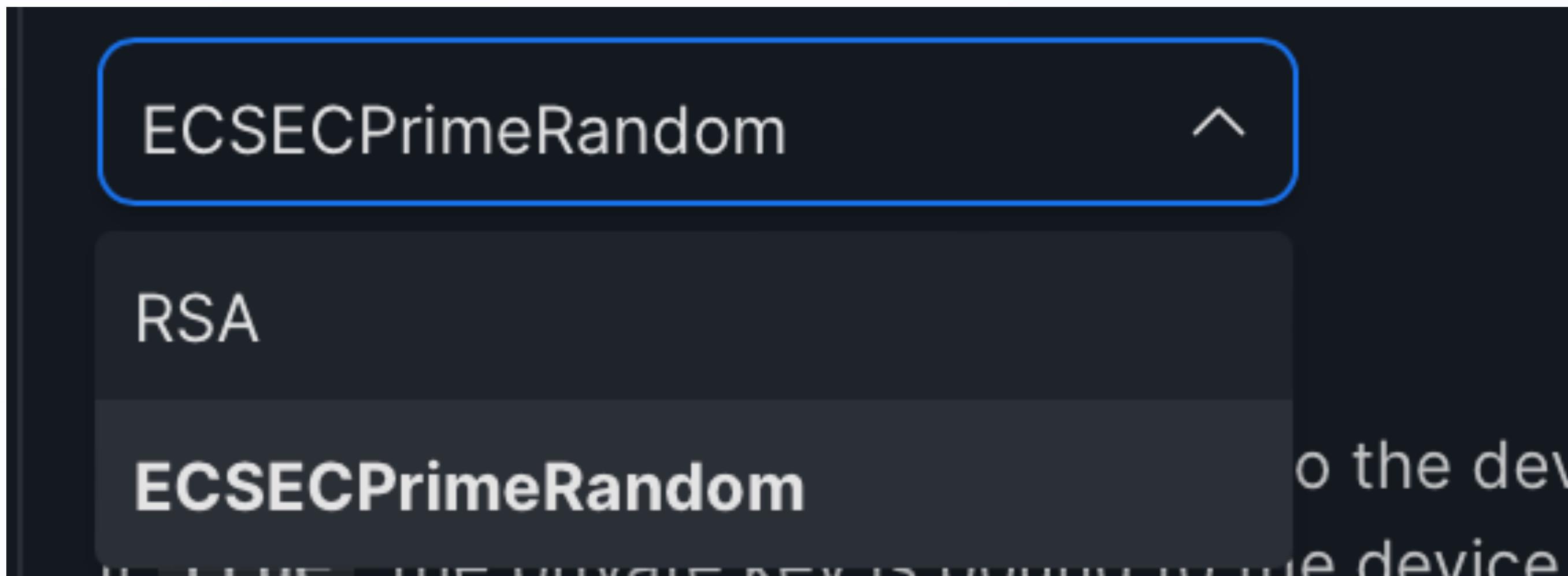
The Secure Enclave generates the key pair, and the private key is cryptographically entangled with a system key. This prevents the system from exporting the private key. If **true**, **Key Type** must be **ECSECPrimeRandom** and **Key Size** must be **256** or **384**. Setting this key to **true** is supported as of macOS 14 on Apple silicon and Intel devices that have a T2 chip. Older macOS versions or other Mac devices require this key but it must have a value of **false**.

True

False



# Jamf ACME with Jamf Pro



**Key Type**  
The type of key pair to generate.

- RSA** : Specifies an RSA key pair. RSA key pairs must have a **Key Size** in the range **[1024..4096]**, inclusive and a multiple of 8, and **Hardware Bound** must be **false**.
- ECSECPrimeRandom** : Specifies a key pair on the P-192, P-256, P-384 or P-521 curves as defined in FIPS Pub 186-4. **Key Size** defines the particular curve, which must be **192**, **256**, **384** or **521**. **Hardware bound** keys only support values of **256** and **384**.

Note that the key size is **521**, not **512**, even though the other key sizes are multiples of 64.

**ECSECPrimeRandom**

**Hardware Bound**  
If **false**, the private key isn't bound to the device.  
If **true**, the private key is bound to the device.

The Secure Enclave generates the key pair, and the private key is cryptographically entangled with a system key. This prevents the system from exporting the private key. If **true**, **Key Type** must be **ECSECPrimeRandom** and **Key Size** must be **256** or **384**. Setting this key to **true** is supported as of macOS 14 on Apple silicon and Intel devices that have a T2 chip. Older macOS versions or other Mac devices require this key but it must have a value of **false**.

True  False

# Jamf ACME with Jamf Pro

**Attest** 

If `true`, the device provides attestations describing the device and the generated key to the ACME server. The server can use the attestations as strong evidence that the key is bound to the device, and that the device has properties listed in the attestation. The server can use that as part of a trust score to decide whether to issue the requested certificate.

When **Attest** is `true`, **Hardware Bound** must also be `true`.

Setting this key to `true` is supported as of macOS 14 on Apple silicon and Intel devices that have a T2 chip. Older macOS versions or other Mac devices require this key but it must have a value of `false`.

True  False

**Allow All Apps Access** 

If `true`, all apps have access to the private key.

True  False

**Key Is Extractable** 

If `true`, the private key of the identity needs to be tagged as “non-extractable” in the keychain.

True  False

ECSECPrimeRandom ^

RSA

ECSECPrimeRandom o the dev

ECSECPrimeRandom

# Recap

- ▶ Understanding certificates
- ▶ Usage of certificates
- ▶ SCEP
- ▶ ACME

# Resources

Slides:



*That's all folks!*