Device Manufacturing and Provisioning with X.509 Certificates in AWS IoT Core

January 2021

This paper has been archived

For the latest version, refer to:

https://docs.aws.amazon.com/whitepapers/latest/device-manufacturing-provisioning/device-manufacturing-provisioning.html

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

This paper has been archived

For the latest technical content, refer to the AWS Whitepapers & Guides page:
https://aws.amazon.com/whitepapers

Contents

Abstract

This whitepaper focuses on onboarding Internet of Things (IoT) devices in AWS IoT Core using unique identities. It covers the different options, challenges, and considerations for manufacturing and provisioning unique X.509 certificates and private keys into devices for certificate-based mutual authentication.

The whitepaper provides device makers with guidance on the appropriate AWS IoT provisioning options, based on the capabilities of their device and manufacturing process. It is not intended to cover Sigv4 and Custom Authorizer authentication methods.

This whitepaper is intended for technical architects, IoT cloud engineers, IoT security architects, and embedded engineers. This whitepaper assumes that the reader understands fundamental Public Key Infrastructure (PKI) and Transport Layer Security (TLS) concepts and terminology.

Introduction

In an IoT deployment, security is often the number one concern from both device customers and device manufacturers. To protect and encrypt data in transit from an IoT device to AWS IoT Core, AWS IoT Core supports TLS-based mutual authentication using X.509 certificates. Device makers must provision a unique identity, including a unique private key and X.509 certificate, into each device. Device makers must also set up the necessary cloud resources on Amazon Web Services (AWS) for each device.

During the different phases of IoT device development and manufacturing, the way that these unique identities are provisioned and onboarded to AWS IoT Core will differ. Device makers are faced with a number of considerations during the lifecycle of an IoT device, including:

- Using a customer-owned Certificate Authority (CA), a third-party CA, or the AWS IoT CA
- Using a hardware security module, such as a secure element
- Cloud resources needed to support the device provisioning process
- Device-level logic to implement on-boarding procedures

This whitepaper explains the complexities of the device manufacturing supply chain, and assists device makers with recommendations on these decisions based on the capabilities of their device and limitations of their manufacturing process.

Resources in AWS IoT Core

For a device to connect to and communicate with AWS IoT Core, AWS IoT Core requires an IoT Thing, Certificate, and IoT Policy.

- **IoT Thing** — AWS strongly recommends that a device is registered as an IoT Thing in the Thing registry. A Thing is a cloud-based representation of a physical device that includes a unique name and static attributes.
- **X.509 Certificate** — Each Thing must have an attached X.509 certificate. The certificate should be unique to a single Thing. The X.509 certificate contains public information including the CA issuer, public key and expiration date. The public key is part of an asymmetrical key pair which includes a private key that is only held on the device.
- **IoT Policy** — An IoT Policy is a document that defines the actions that the device is authorized to perform. The IoT Policy must be attached to the X.509 Certificate. A Policy can be shared among many devices with the use of policy variables.

Resources on the Device

For the device to connect to AWS IoT Core using TLS-based mutual authentication, the device needs to be provisioned with the Amazon Trust Services server certificate, an X.509 certificate, a private key, and in some cases, the issuing CA for the device's client certificate.

- **X.509 Certificate** — The same X.509 certificate that is present on AWS must also be present on the device. This certificate is presented during the TLS handshake with AWS IoT Core.
- **Private Key** — The private key of the device is asymmetrically paired with the public key that is presented with the X.509 certificate. The private key is ideally generated on the device using a True Random Number Generator and should never be exported from the device.
- **Signer Certificate Authority** — The device will need to send the X.509 certificate's issuing CA in the first TLS connection if the Just in Time device onboarding flow is used. Subsequent connections do not require the issuing CA certificate.
- **Server Certificate** — The Amazon Trust Services (ATS) server certificate is used by the device to verify that it is connecting to a genuine AWS IoT ATS Endpoint.

Device Provisioning During Development

In the early phases of an IoT project, a small number of devices will be provisioned to AWS IoT for development and testing purposes. During this phase, developers often choose convenience over security and scalability.

AWS IoT provides the ability to create all necessary resources to provision a single Thing from the AWS Management Console, the AWS IoT Control Plane APIs, or the AWS Command Line Interface (AWS CLI). The certificate is issued by an AWS CA, and the private key is generated on the AWS Cloud. The certificate, private key, and public key are downloaded onto the developer's local machine.

The developer is responsible for manually provisioning the certificate and private key from their local machine to the device. The certificate and private key are either added to the device's file system or can be written directly into the device's firmware code.

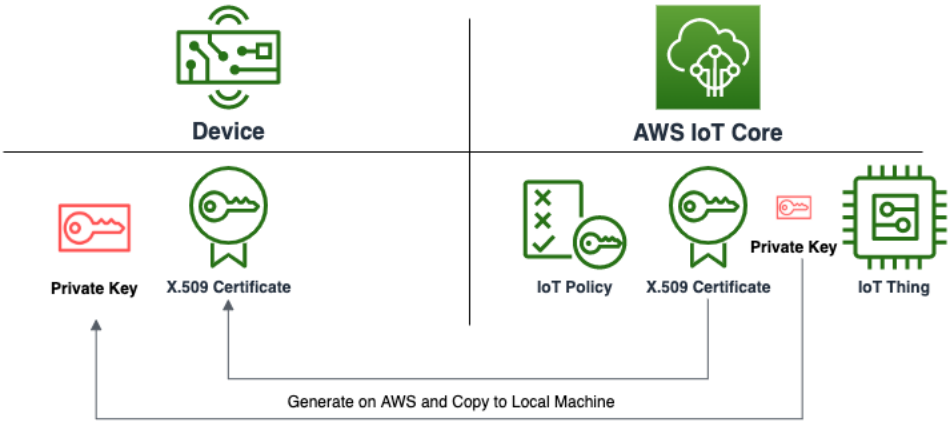

Figure 1: Process to provision devices during development using AWS IoT Core

This process should never be used in production, as the developer's local machine could be compromised by social engineering, user error, or a weak password.

Security risks are decreased because the devices are not deployed in the field. The devices are tightly controlled by a developer or in a lab environment. A compromised device can be re-programmed, or the certificate can be deactivated.

The process of creating all resources in the AWS Cloud and copying the necessary keys to the device is manual and requires programming expertise, and files need to be programmed into each device by the developer. In this scenario, most operations are performed in the lab environment, but the process is not scalable when entering the pilot or production phase of an IoT project. To scale an IoT project, there is typically a much more complex supply chain.

The Device Manufacturing Supply Chain

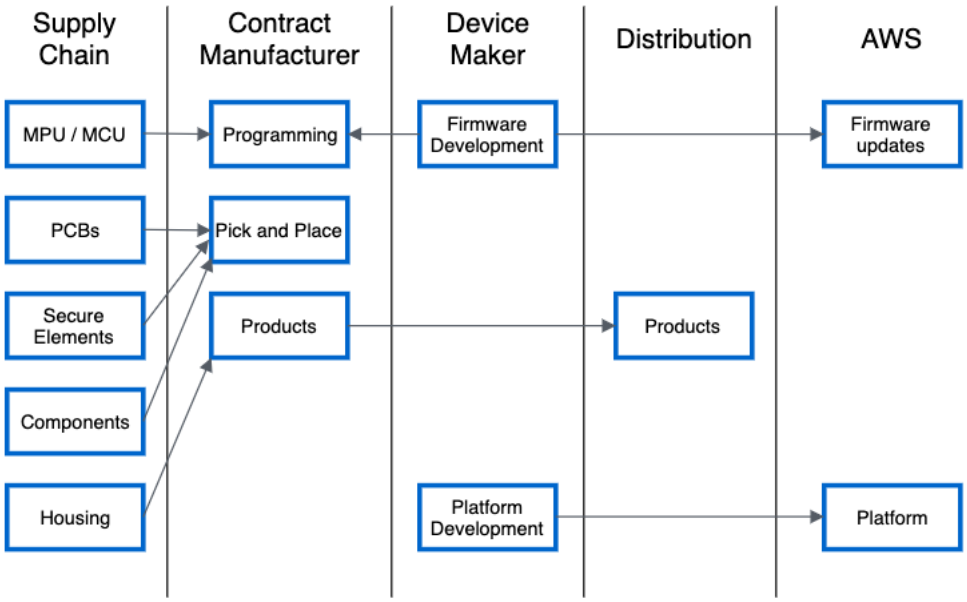

Figure 2: The IoT device manufacturing process

As an IoT project moves from a development phase to production, a supply chain is created between the device maker through to the customer.

Device makers specify the components, form factor, and functionality of a device. Device makers are responsible for product design, developing hardware and software for products, developing and maintaining AWS Cloud applications, provisioning templates, policies, and resources, and the sales and marketing of the product. Most device makers do not have the ability to physically manufacture a large number of the product, and must outsource this manufacturing to a contract manufacturer.

During the prototyping stage, device makers may use high-mix, low-volume contract manufacturing to rapidly produce engineering samples in low volumes. When moving from the prototyping to production phase, a high-volume contract manufacturer is used to take advantage of the economy of scale.

Low-mix, high-volume contract manufacturers have the production line, tooling, and processes in place to produce a large volume of devices. Contract manufacturers build devices to the specification provided by the device maker. This specification includes printed circuit board (PCB) schematics, a bill of materials, and the firmware that is loaded onto the device. Contract manufacturers place components onto PCBs (pick and place), program the processors with firmware and credentials provided by the device makers, and package the devices into a final product. Products are then provided to distribution channels to fulfill sales. Contract manufacturers must source the individual components to build the final product from their supply chain.

Vendors in the contract manufacturing supply chain produce individual components that are used on the device. Examples of components include microcontrollers, processors, secure elements, and connectivity modules. Component vendors may rely on distributors to fulfill the supply chain to contract manufacturers.

After the device is built, it may be sent to the device maker for maintaining stock levels or engineering samples. The devices may also be sent to distributors who sell the device and fulfill the supply chain to end customers.

Provisioning Timing in the Device Manufacturing Supply Chain

When creating the device manufacturing supply chain, device makers must specify when devices receive their unique X.509 certificate and private key.

Device makers may choose to provision devices with unique credentials in firmware. In this scenario, a unique firmware is generated for each device that contains the X.509 certificate and private key in code. Each unique firmware image is sent to the contract manufacturer. Contract manufacturers must have the ability to program each individual firmware image onto the device's processor at manufacture time. In this scenario, the device maker maintains the most control over the supply chain and has the opportunity to pre-register each device on AWS IoT. This adds considerable complexity to the contract manufacturer.

For the la… …the AWS

http… …pers

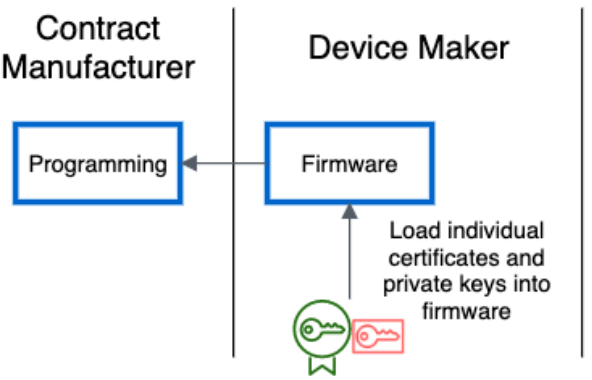

Figure 3: Provision credentials into firmware

Contract manufacturers may offer the ability to perform customization during the manufacturing process. In this scenario, a single firmware image, known as a golden image, is provided to the contract manufacturer and the manufacturer is responsible for delivering the credentials to the device. The firmware must have additional logic that allows the device to accept credentials over an open interface, such as Secure Shell. (SSH), Network File System (NFS), or over a serial connection, and store those credentials to a secure place on the device. Security credentials are exposed in the

manufacturing environment, and PKI is handled by the contract manufacturer. It is important that the provisioning process is performed in a secure environment by trusted individuals.

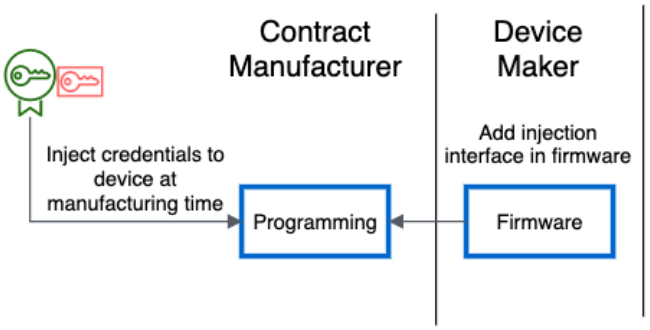

Figure 4: Inject credentials at manufacturing time

Introducing customization for each device at manufacturing time can add valuable time to produce each device and additional logistical overhead as the manufacturer must track that customization for each device produced. This leads to increased cost per unit to the device maker charged by the contract manufacturer.

Device makers may choose to add a hardware security module (HSM), such as a secure element or trusted platform module (TPM) on their device. Hardware security module vendors have processes in place to generate private keys and sign X.509 certificates in the vendor's secure facility. This allows device makers to provide a single standard firmware image to the contract manufacturer with logic to communicate with the HSM. The contract manufacturer is responsible for placing the HSM onto the PCB, but the credentials are not exposed and no extra processes are necessary to provision the credentials.

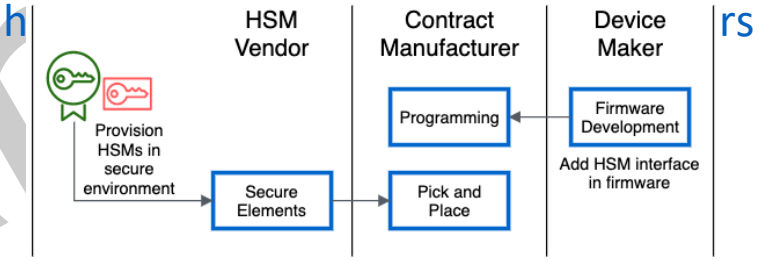

Figure 5: Hardware Security Module vendor provisions credentials to a secure element

Third parties offer value-added services such as secure programming of applications and credentials. Device makers may use a trusted distributor or third party in the supply chain to provision credentials into their devices. This may happen in the supply chain before a device is assembled by the contract manufacturer, or after the devices have been produced in the supply chain to fulfill sales to end customers.

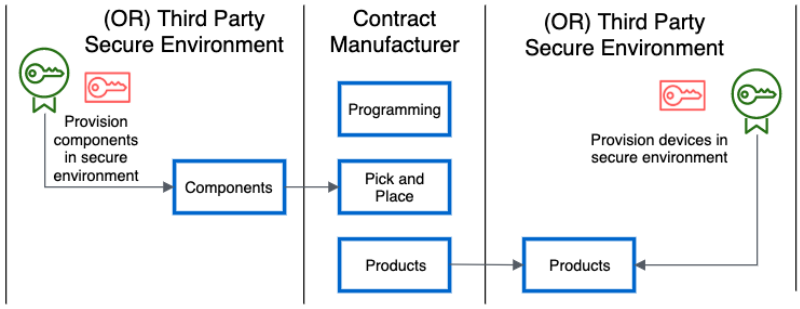

Figure 6: Third party vendor provisions credentials in a secure environment

Devices Not Provisioned in the Supply Chain

Manufacturers, component vendors, or distributors may have minimum order quantities before they provide customization services. Introducing customization or a pre-provisioned hardware security module may be cost prohibitive for low volume devices. In consumer products, sometimes it is not known where the device should be provisioned at the time of manufacturing. In these scenarios, devices come out of manufacturing with no unique credentials. Devices can be provisioned after they are sold and deployed in the field using a service called fleet provisioning.

Managing a Certificate Authority and the Chain of Trust

Certificate Authorities issue and sign X.509 certificates. Device makers must decide whether they would like to use AWS IoT as the CA, their own PCA, or a third-party CA.

AWS IoT provides the ability to generate X.509 certificates and private keys on the cloud. The X.509 certificates are signed by an AWS IoT CA and are preregistered in the device maker's AWS IoT registry at creation. Once created, the device maker must download the certificate and private key and deliver the certificate and private key to the device during manufacturing.

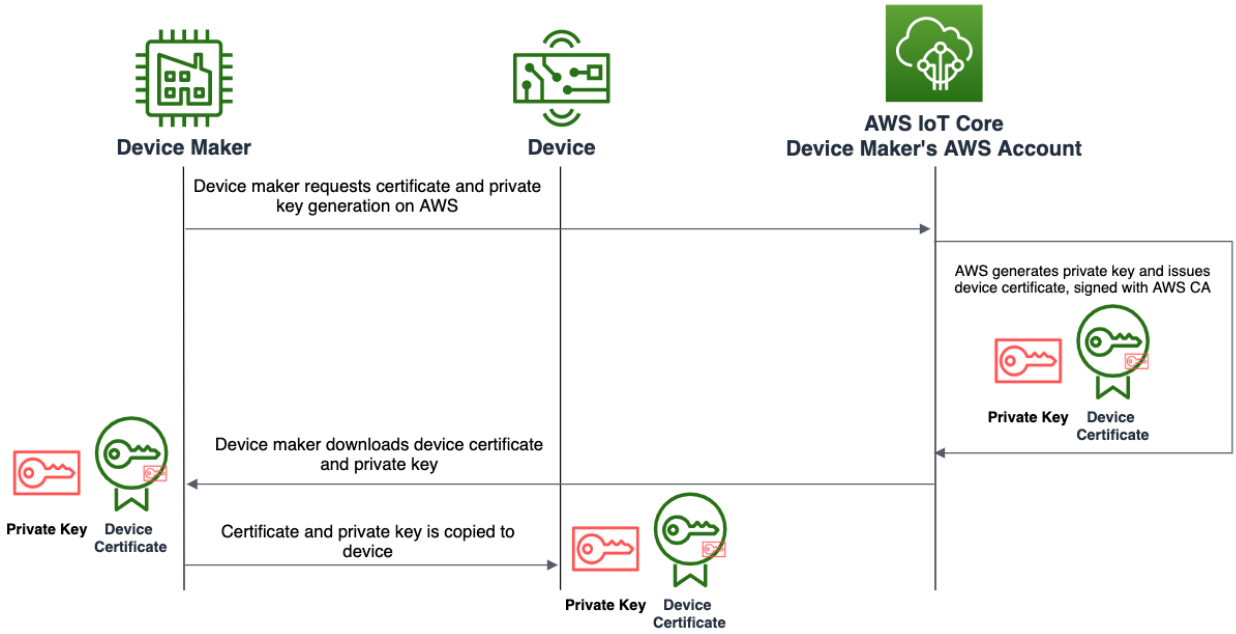

Figure 7: X.509 certificate and private key generated on AWS

If the device already has a private key onboard, a Certificate Signing Request can be sent to AWS to sign the certificate without exposing the private key on the device.

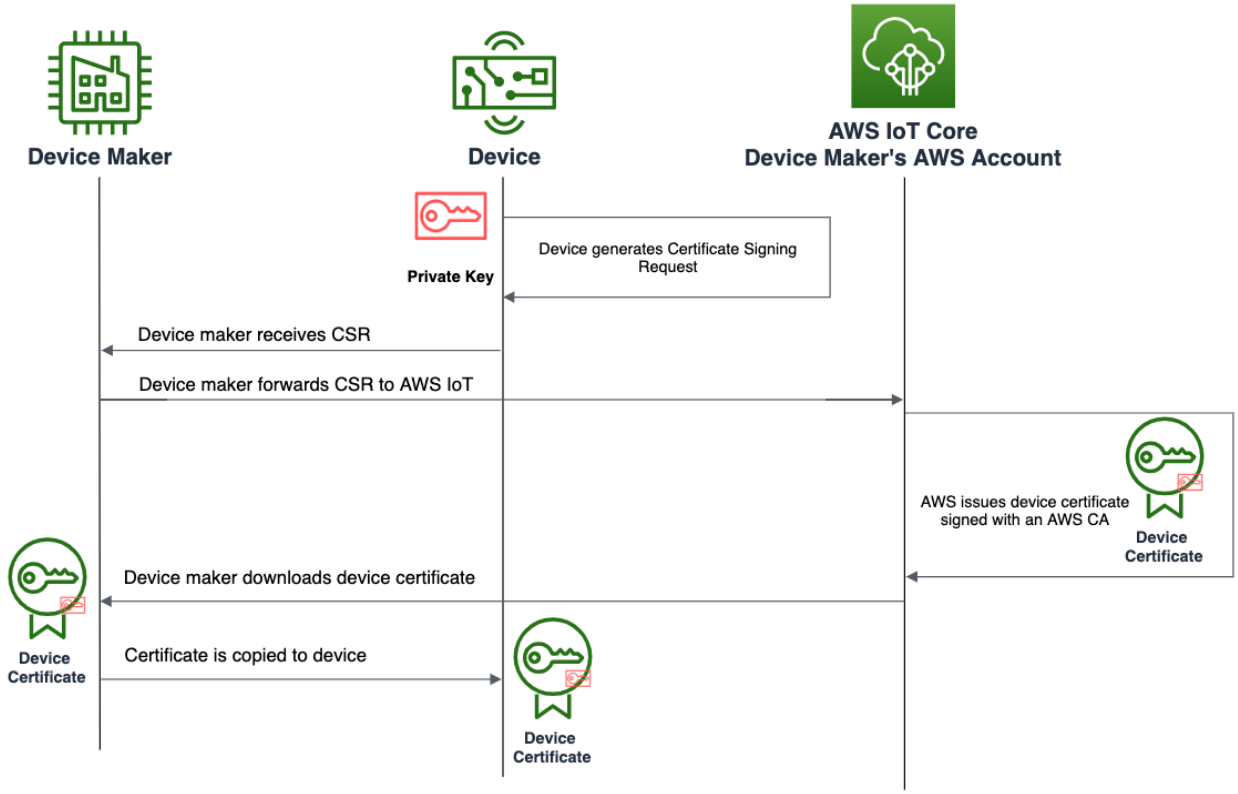

Figure 8: Certificate Signing Request made by device to AWS

The certificates and private keys must be included in the firmware of each device, or provided to the device at runtime. Loading certificates onto devices is protected under the AWS Shared Responsibility Model, so the device maker does not need to implement their own controls on the CA. The device maker is responsible for the authorization policies granted to users of the AWS Account to ensure only authorized users can generate new certificates.

If the device maker must maintain control of the CA and Public Key Infrastructure, AWS IoT provides the option to use a customer owned CA. Devices interact directly with the CA through a secure network channel to create a Certificate Signing Request during the manufacturing process. If the device cannot access the CA directly, the certificates and private keys can be pre-generated and loaded onto the device in firmware, on a hardware security module, or delivered over a secure local connection in the manufacturing process. The certificates must be registered on AWS IoT before the device is able to connect for the first time.

Large enterprises typically have their own self-signed root CA. A self-signed CA provides the greatest level of flexibility and control over the Public Key Infrastructure. It is necessary to employ strict security protocols to protect the CA from being compromised, such as a key signing ceremony and physical access control. When the certificate authority is self-signed, there is typically a chain of one or more intermediate signer certificates. This allows the device maker more strict control of the certificate

revocation list by allowing an intermediate certificate to be revoked without affecting the rest of the certificate infrastructure. The device certificate's issuing signer CA must be registered with AWS IoT.

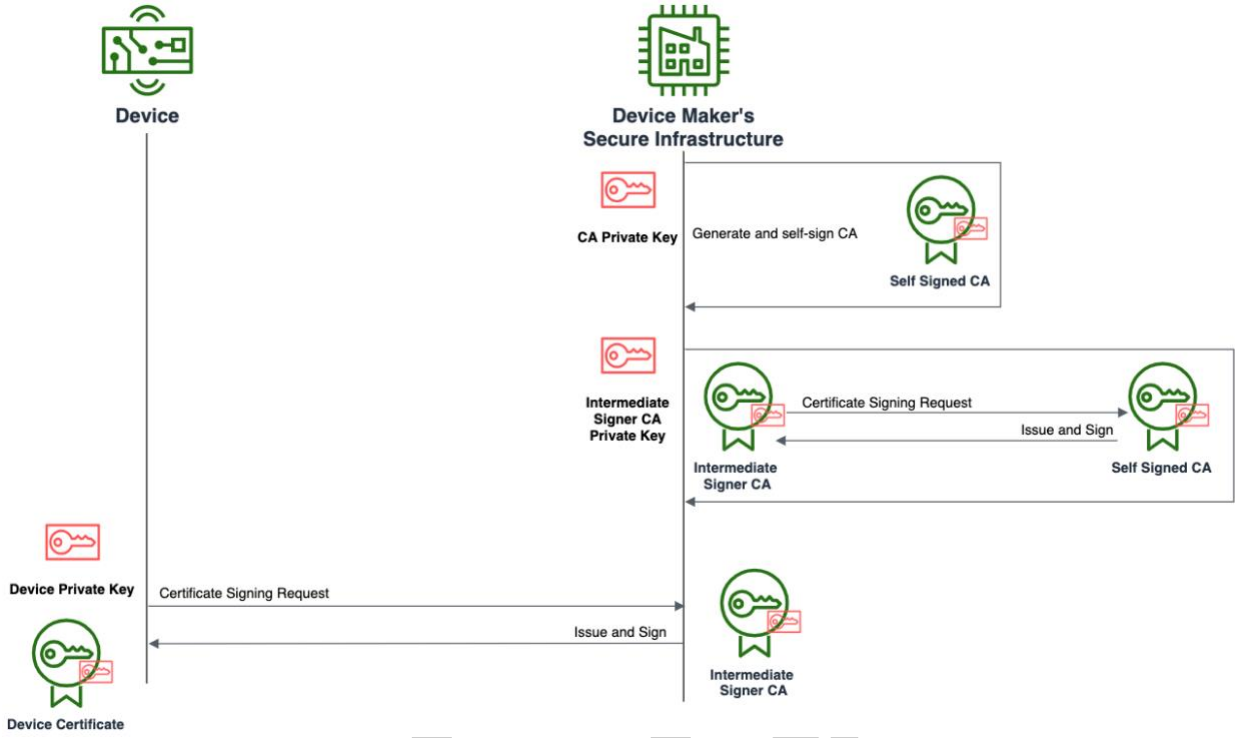

Figure 9: Self Signed Certificate Authority and Intermediate Signer CA infrastructure

AWS Certificate Manager (ACM) is a managed service on AWS that can generate and sign X.509 certificates in the cloud. The flexibility of ACM allows customers to bring their own CA and perform certificate signing operations on AWS. AWS protects the physical infrastructure where the CA is held on the ACM service, and the device maker is responsible for enacting appropriate policies for users that have access to the ACM service in their account.

If the device maker does not want to maintain its own CA, but still wants to control the Public Key Infrastructure for their assets, third-parties offer CA services. These CA service companies will generate an intermediate signer CA for the device maker that is customized to the device maker's specification, or may sign certificates from their own root CA. Third party CAs give the device maker the ability to generate and sign X.509 certificates, but the third party maintains the physical security of the CA. Hardware security module vendors typically offer this service to pre-provision their modules before shipping them to the contract manufacturer.

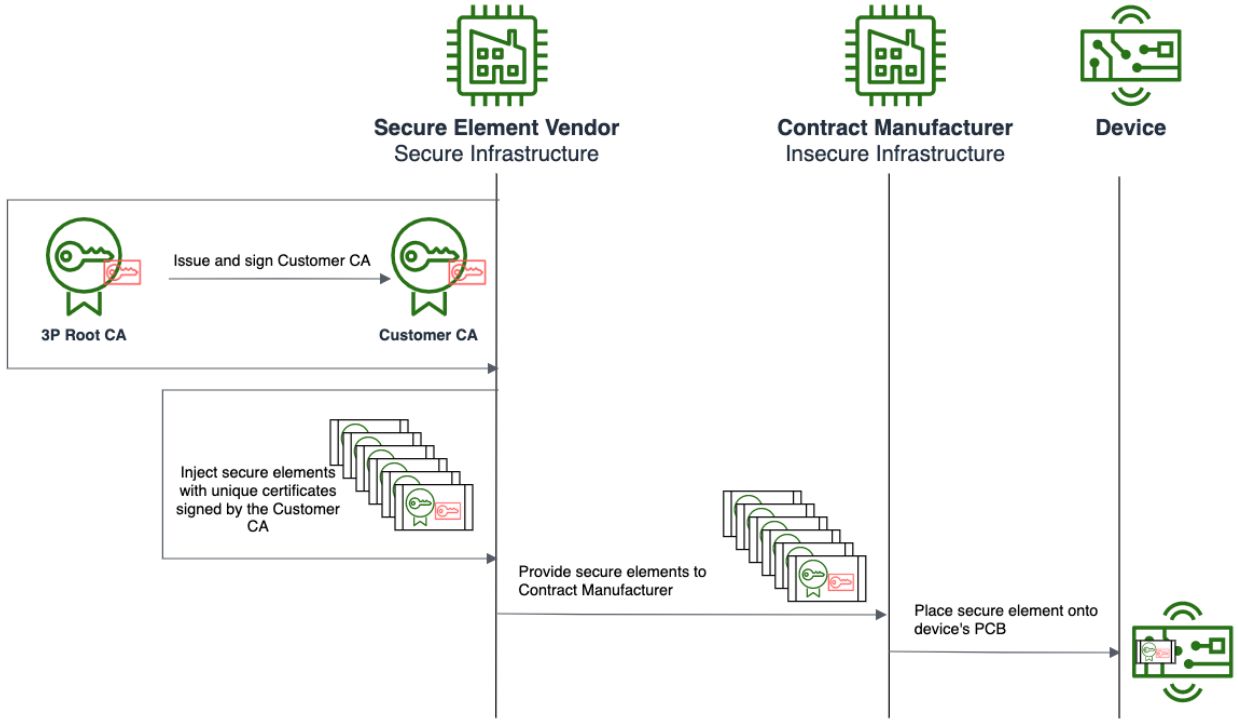

Figure 10: Third party Certificate Authority with Hardware Security Modules

AWS IoT Provisioning Options

AWS IoT provides options to provision and onboard a large number of devices based on the capabilities of the device and if the devices have their unique X.509 certificate and private keys on them before being sold to the end customer.

If the manufacturing chain allows the device maker to provision unique credentials into the device at manufacturing time or in distribution, device makers can use Just in Time Provisioning, Just in Time Registration, or Multi-Account Registration.

If it is not possible to deliver credentials to the device before the device is sold to the end customers, device makers may use Fleet Provisioning to onboard their devices.

Just in Time Provisioning (JITP)

Devices that use JITP have certificates and private keys present on the device before onboarding to AWS IoT. The certificates must be signed with the customer's designated CA, and that CA must be registered in AWS IoT. The customer must know which account the device will connect to before provisioning.

Setup

Using JITP, the device connects to AWS IoT, and the certificate's signature is verified against the registered CA. After verification, a provisioning template registers the Thing, certificate, and assigns a policy to the device. The device maker is responsible for registering the signer CA and attaching a provisioning template to the CA.

Device Logic

When the device connects to AWS IoT Core for the first time, the device certificate, and the signer CA that is registered with AWS IoT must be sent during the TLS handshake. The TLS handshake will fail at the first connection. This happens because the certificate has not been pre-loaded into the AWS IoT account. The device-supplied certificate is registered and activated in AWS IoT during the provisioning process. The device must have logic to reconnect to AWS IoT after a short time period. If the provisioning operation has succeeded, the device will connect to AWS IoT successfully.

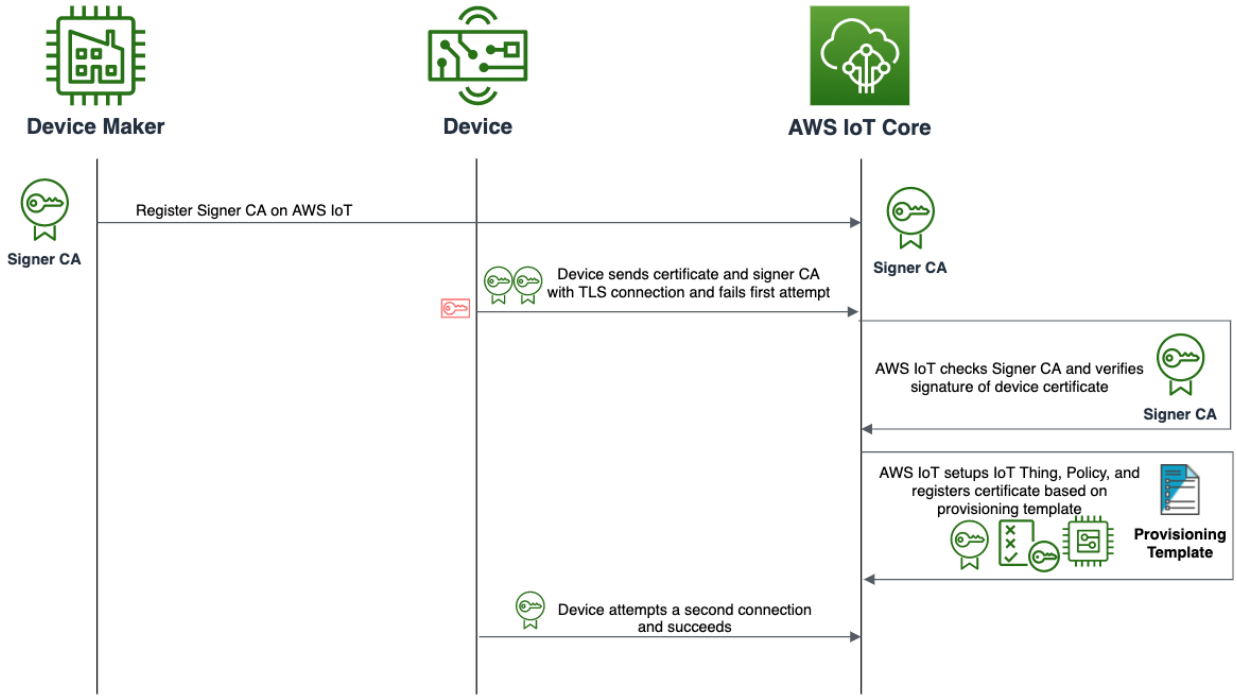

Figure 11: Just in Time Provisioning process

Just in Time Registration (JITR)

JITR should be used when additional custom logic is required when the device is registered on AWS IoT. Like JITP, certificates and private keys must be present on the device before onboarding and the signer CA must be loaded into the AWS Account before the device onboards.

Setup

When the device first connects to AWS IoT, the certificate's signature is verified against the Certificate Authority and the certificate is registered in an inactive state. AWS IoT generates a Lifecycle event on the MQTT topic `'$aws/events/certificates/registered/<caCertificateID>'`. The device maker sets up an IoT Rule that triggers an AWS Lambda function whenever a message is published on that topic.

The Lambda function can perform actions such as secondary validation against an Allow list or Certificate Revocation List, register a device to a particular user on the cloud platform, or trigger additional onboarding workflows. The Lambda function sets the certificate's state to Active after additional validation is completed. The Lambda function should also register the IoT Thing and set up the Policy.

Device Logic

Just like JITP, the device will fail to make a connection on the first connection to AWS IoT. The device must contain logic to reconnect to AWS IoT after the first failed connection. If the Lambda function activates the certificate, the second device connection will succeed.

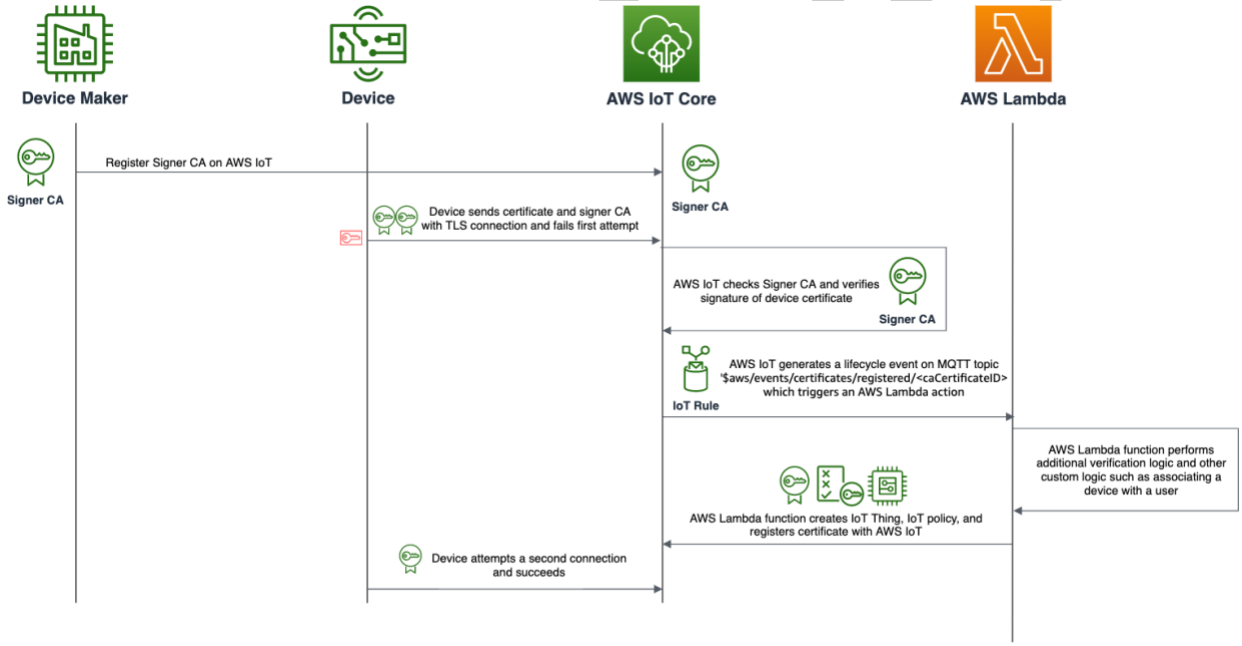

Figure 12: Just in Time Registration process

Use Cases for Just in Time Provisioning and Registration

Just in Time Provisioning and Just in Time Registration are used when the device maker knows at manufacturing time in which AWS Account and Region the device will

be provisioned. The CA must be registered in the account in which the device will be onboarded before the devices attempt a first connection. The Just in Time process requires a one-time setup, and will scale to millions of devices.

Multi-Account Registration (MAR)

Devices that are provisioned to AWS IoT Core using MAR need unique certificates and private keys present on the device before being onboarded. Certificates are signed with a CA, but that CA is not required to be registered with AWS IoT. Certificates can be registered in any region and in any account that the device maker has access to.

Setup

Device makers are required to manually set up the necessary resources in AWS for each device. The resources are set up before the device connects to AWS IoT for the first time.

Some vendors provide hardware security modules that are pre-provisioned with a private key and certificate. The certificate is signed with the vendor's own CA, and the vendor provides a manifest of certificates to the device maker. The device maker is responsible for registering the certificates in each account and region using the AWS Management Console, AWS IoT Control Plane API, or the AWS CLI.

Device Logic

The device's TLS stack must support the Service Name Indicator (SNI) extension, and the AWS IoT Core endpoint is passed in the SNI string. No additional device logic is required to perform MAR provisioning.

Use Cases

MAR is used when the device maker requires flexibility in where their devices should be provisioned to. Device makers may have multiple AWS Accounts used for sandbox, testing and production. Certificates can be pre-registered in each AWS Account and the device can connect to different accounts throughout its lifecycle. Devices can be sold globally, and the certificate can be registered in multiple AWS Regions as well as multiple AWS Accounts within the same Region. MAR allows the device makers to provide the public X.509 certificate to an IoT service provider so that the device can be onboarded to the service provider's account, or the device may connect to multiple accounts to send data to each endpoint. MAR is also used when device makers do not own the signing Certificate Authority and are not able to register the CA with AWS IoT.

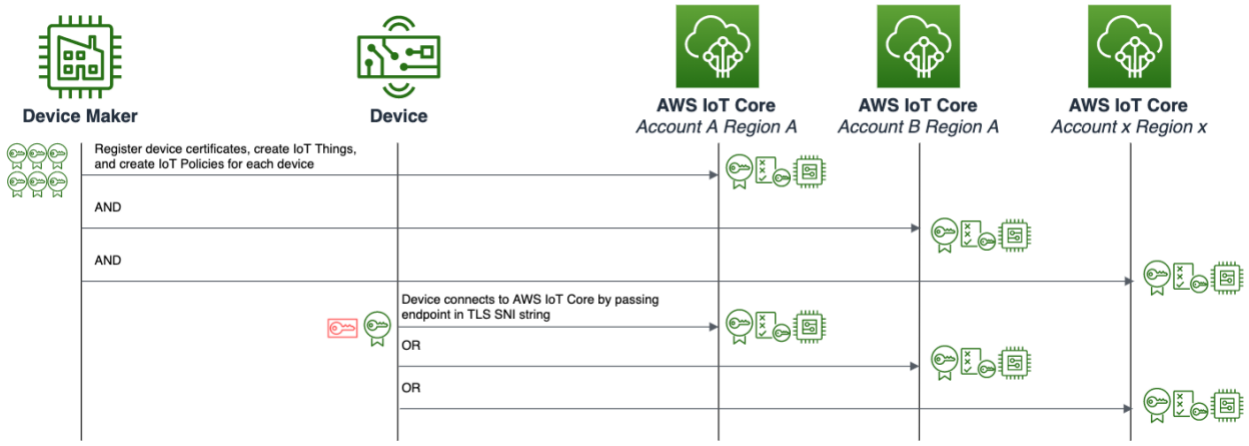

Figure 13: Multi Account Registration (MAR) process

Fleet Provisioning

There are many cases when provisioning unique credentials to a device at manufacturing time is prohibitive due to technical limitations, cost, or application specific limitations. Devices that are not customized from the manufacturer are sold to customers without a unique identity. Fleet Provisioning provides two ways to provision devices with unique credentials after they are delivered to end customers: by Trusted User or by Claim.

Fleet Provisioning by Trusted User

AWS IoT provides an application programming interface (API) that allows mobile applications to generate temporary certificates and private keys. The device leaves the manufacturing facility with no unique credentials, and only Trusted Users are able to provision the device with its unique credentials.

An installer uses a mobile application and authenticates with AWS. Using the Trusted User APIs, the installer receives a temporary X.509 certificate and private key that is valid for five minutes. Using the mobile application, the credentials are delivered to the device. The device connects to AWS IoT and exchanges the temporary credentials for a unique X.509 certificate signed with the AWS CA and a private key. During this workflow, the AWS resources including Thing name, Policy, and Certificate are set up in the AWS Account.

Setup

Device makers that use the Trusted User flow must develop and maintain a mobile application that exercises the Trusted User APIs. The Fleet Provisioning template must be set up and maintained in AWS IoT by the device maker. Optionally, an AWS Lambda

function can be used to provide additional authentication steps during the provisioning process.

Device Logic

Devices must have the ability to accept temporary credentials over a secure connection such as Bluetooth Low Energy, WiFi, or USB. Devices must implement the logic necessary to publish and subscribe to Fleet Provisioning MQTT topics, accept the permanent credentials, and write the credentials to secure storage.

Use Cases

The credentials are never exposed to the manufacturing supply chain. Fleet Provisioning by Trusted User is the recommended approach when a high degree of security is needed, when the manufacturing chain is not trusted, or it is not possible to provision devices in the manufacturing chain.

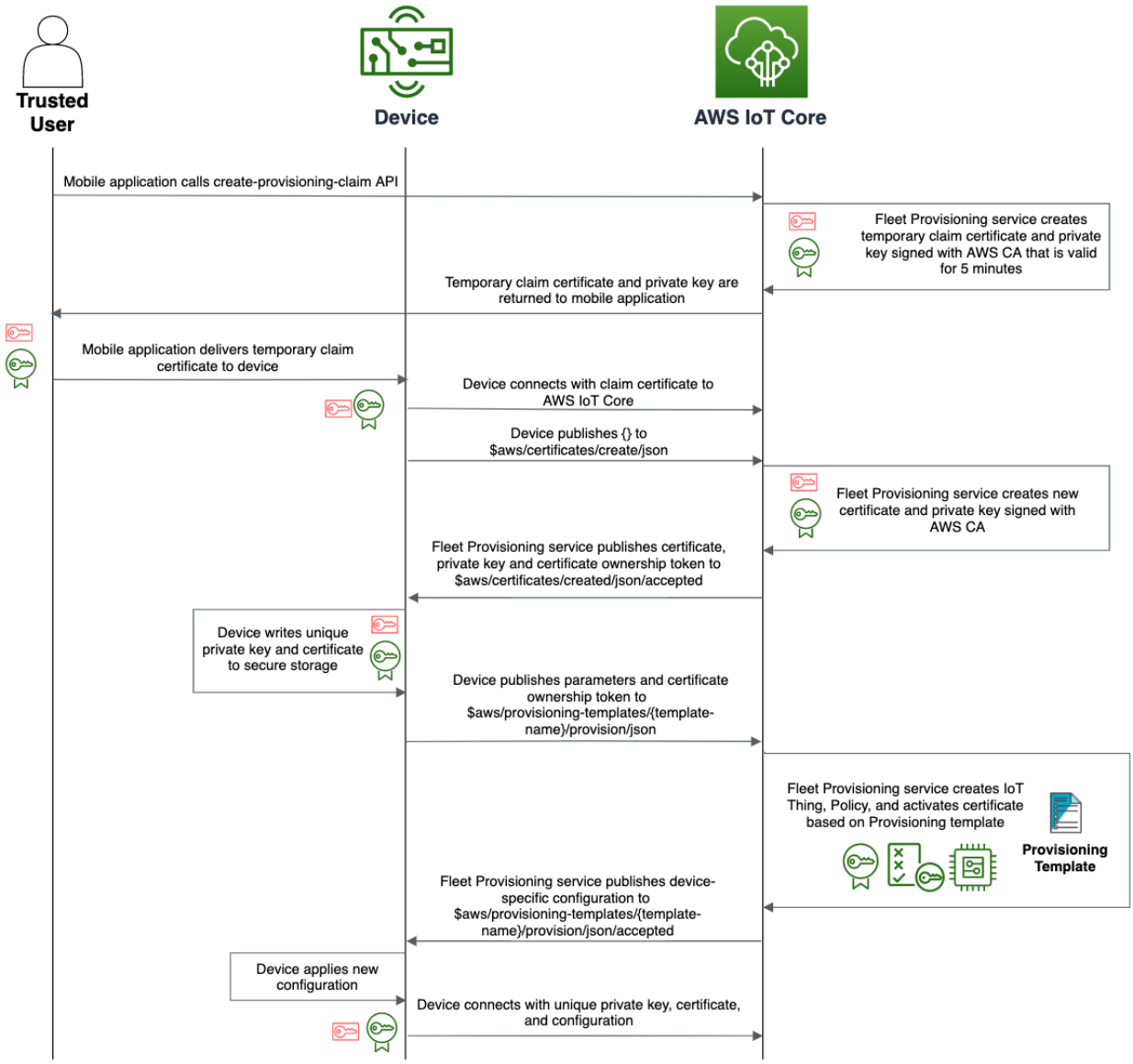

Figure 14: Fleet Provisioning by Trusted User process

Fleet Provisioning by Claim

Some devices do not have the capability to accept credentials over a secure transport, and the manufacturing supply chain is not equipped to customize devices at manufacturing time. AWS IoT provides a path for these devices to receive a unique identity when they are deployed.

Device makers must load each device with a shared claim certificate in firmware. This claim certificate should be unique per batch of devices. The firmware containing the claim certificate is loaded by the contract manufacturer without the need to perform any customization. When the device establishes a connection with AWS IoT for the first

time, it exchanges the claim certificate for a unique X.509 certificate signed by the AWS certificate authority and a private key. The device should send a unique token, such as a serial number or embedded hardware secret with its provisioning request that the Fleet Provisioning service can use to verify against an Allow List.

Setup

Device makers that use Fleet Provisioning by Claim must maintain a Fleet Provisioning template and AWS Lambda function with additional verification logic. The claim certificate must be protected and frequently audited to prevent misuse.

Device Logic

Devices must implement the logic necessary to publish and subscribe to Fleet Provisioning MQTT topics, accept the permanent credentials, and write the credentials to secure storage.

Use Cases

Fleet Provisioning by Claim should be used only when it is not possible to provision unique credentials in the device, and the manufacturing supply chain is secured by trusted individuals from contract manufacturer to the end customer. Devices that enter insecure channels such as a third-party distributor should not use Fleet Provisioning by Claim.

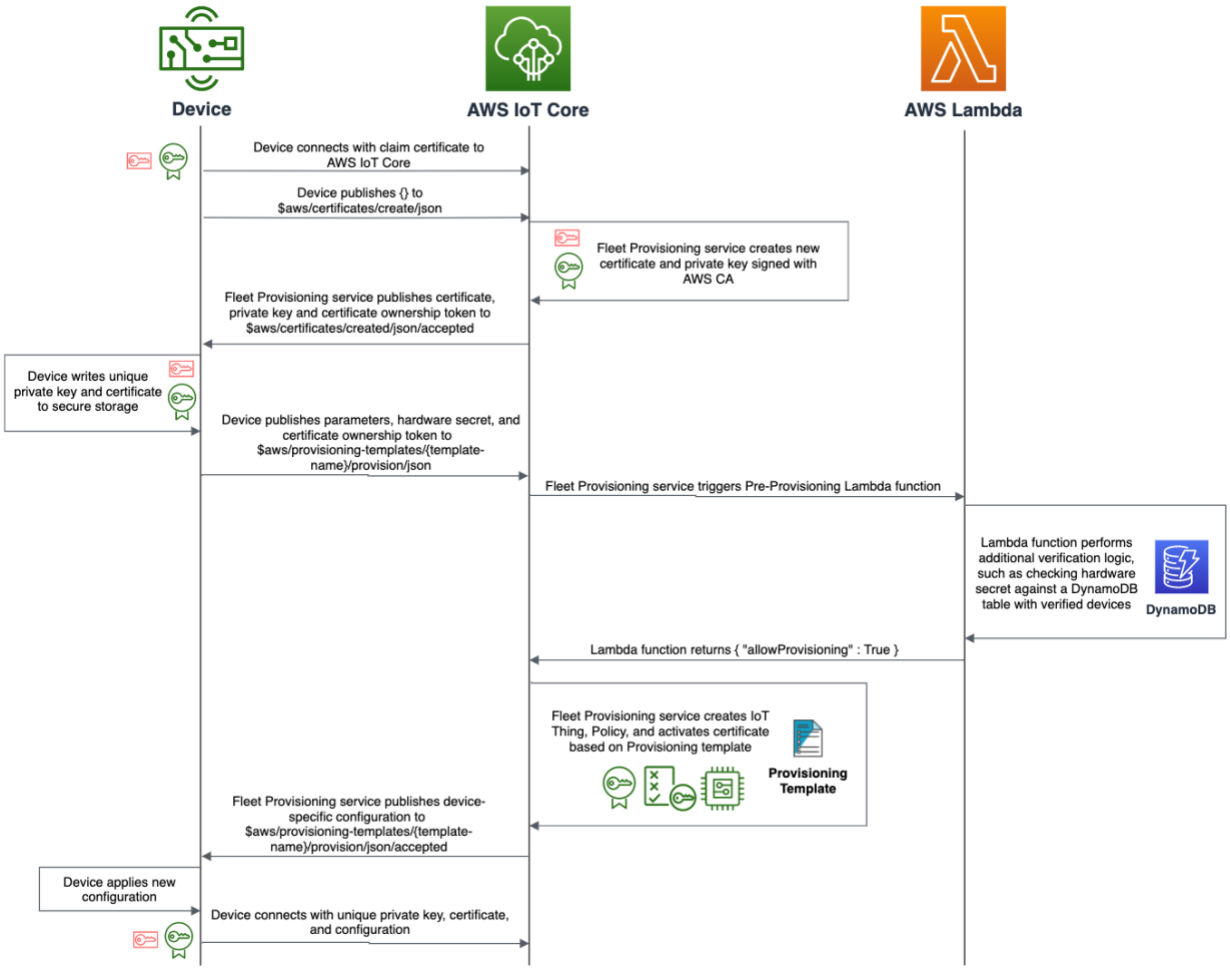

Figure 15: Fleet Provisioning by Claim process

Whitepapers & Guides page:
https://aws.amazon.com/whitepapers

Conclusion

Moving an IoT project from proof of concept to a large-scale deployment is a complex challenge. Device makers must make multiple decisions that fundamentally impact the device's capabilities, level of security, bill of materials, and product and operational cost. Production and deployment decisions must be considered early on in the development process because they will impact manufacturing and deployment costs.

Device makers must consider the following when starting their IoT project:

- Who will maintain the Public Key Infrastructure for their project?
- What capabilities should they build into their device?
- What capabilities do their contract manufacturer have to provide customization at manufacturing times?

This whitepaper has outlined the options that AWS IoT provides for device makers as they answer these questions. No matter the capabilities of the device and manufacturing process, level of trust in the manufacturing supply chain, or security requirements, AWS IoT provides options for device makers to onboard their devices at scale in a secure way.

Contributors

Contributors to this document include:

- David Walters, Senior Partner Solutions Architect, IoT
- Michael Schy, Senior Partner Solutions Architect, IoT
- Ashok Bhaskar, Senior Partner Solutions Architect, IoT
- Tim Mattison, Principal Partner Solutions Architect, IoT
- Alok Jha, Senior Product manager, AWS IoT Core

Document Revisions

Date	Description
January 2021	First publication