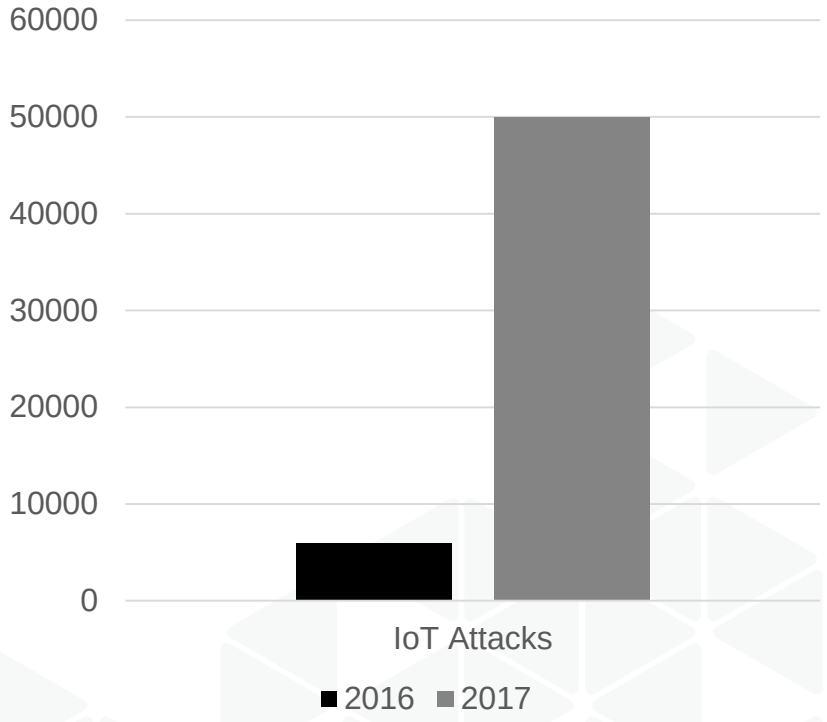
SMART | CONNECTED | SECURE

Incredible growth in IoT security attacks

600%



*Symantec 2018 Internet Security Threat Report

The Problem with all “known” protocols

Protocols like TLS and the on-boarding to cloud services are published and widely used

- They make an excellent target for hackers because any success impacts large populations
  - ***This means the value of a successful hack is incredibly high!***
  - Because of this value, countries, corporations, and the most talented individuals all target these protocols

New successful hacks and threats are identified every month

- They rarely attack the crypto primitives
- They attack what’s around them, hence the term “side channel” attack

SMART | CONNECTED | SECURE

Embedded Security for TLS and Cloud Services

MICROCHIP

One thing in common

Unique Identities

The one thing all these cloud services have in common is the requirement that each attached device needs a ***unique identity***

- All devices need to be *provisioned – add secrets and/or credentials which create and serve to authenticate an identity*

This ***unique identity*** is used to authenticate the device(s) and allow access to the service(s)

- Some cloud services a chain of trust to establish identity
- Others use require the OEM to submit each identity to the service before that device asks for access.
- Some offer both options

Transport Layer Security (TLS)

TLS evolved from Netscape's Secure Sockets Layer (SSL)

- TLS is essentially SSL 3.1 but the terms "SSL" and "SSL/TLS are still used.

The TLS protocol provides Authentication, Integrity and Confidentiality

- It's the most widely deployed security protocol used today
- Used by web browsers and anything requiring secure data exchange
  - File transfers, VPN, Instant Messaging (e.g. MQTT), VoIP, etc…

TLS can be "Hardened" by using H/W enforced embedded security

What Provisioning does TLS require?

To be secure, any device needs a unique trusted identity

- Provisioning creates this unique trusted identity

TLS requires a chain of trust supported by the Public Key Infrastructure (PKI)

- Public/Private key pair
- Digital Signatures and Certificates for
  - Authentication
  - Integrity
  - Confidentiality

Any device connecting through TLS needs to be provisioned

- The only choice is; where will you place your keys and secrets ?
  - Out in the open?
  - Locked in a hardware secure device?

Hardened TLS Overview

SMART | CONNECTED | SECURE

Functions on Host MCU

- Protocol handling
- Cypher suite negotiation with server
- Data stream encryption/decryption (AES)
  - For large messages

Functions off-loaded from TLS stack to Secure H/W

- Server authentication (ECDSA)
- Client authentication (ECDSA)
- Provides Client certificate chain
- Generate all needed keys (ECDHE/KDF)
- AES encryption for small messages

- Smaller (D)TLS memory footprint on host MCU
- ***Faster and lower power*** execution vs. software implementation
- TLS software can not leak keys under any circumstances
- No need to inject keys & certificate into host MCU application during manufacturing

SMART | CONNECTED | SECURE

The pillars of hardware authentication

1

Isolate private keys from users and software

- Humans are the most unpredictable security risk
- Patches reveal the software weakness to an attacker. It can take too long to patch all IoT hardware which gives an attacker time to penetrate the system.

2

Isolate private keys and critical crypto-primitives from software

- Protect everything to do with Authentication, Integrity and Confidentiality in a hardware secured vault

3

Isolate key manipulation from the manufacturing phase

- Not only from the supply chain equipment but also from the operators in the supply chain.

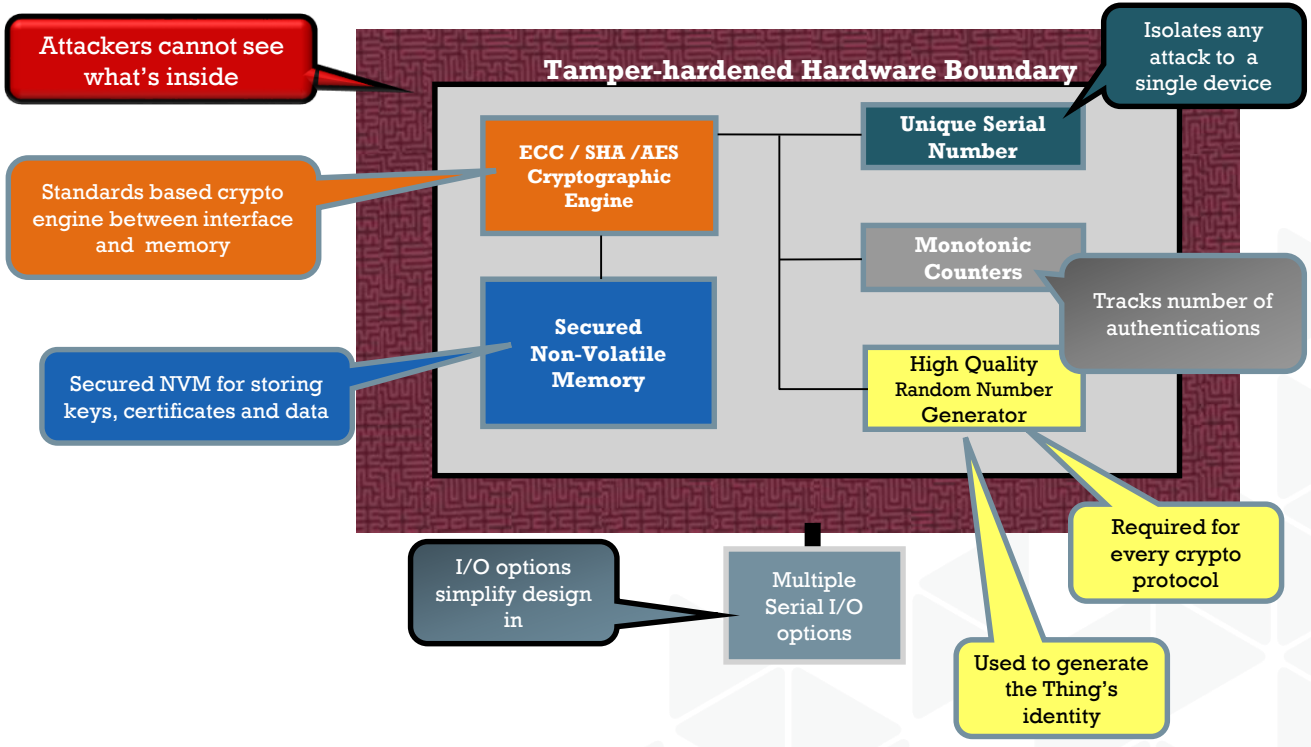
4

Isolate keys from microcontrollers

- Do Not store the private key in unprotected MCU flash memory

MICROCHIP

The Building Blocks to Any Secure Solution



How security is enhanced for …

SMART | CONNECTED | SECURE

MICROCHIP

What Provisioning does AWS require?

AWS uses TLS – *Mutual Authentication*

- TLS requires a chain of trust supported by the Public Key Infrastructure (PKI)
  - Public/Private key pair
  - Digital Signatures and Certificates for
    - Device Authentication
    - Message Integrity
    - Establish rules for Confidentiality

Any device connecting to AWS needs to be provisioned before it can connect

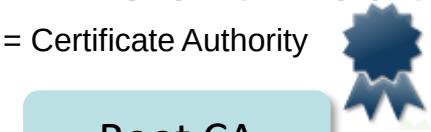
AWS, Root of Trust

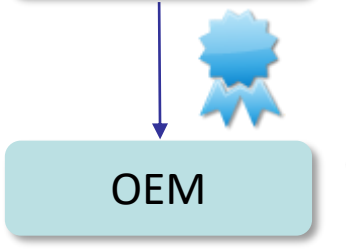
BYOC = Bring your own certificate

- AWS IoT allows certificates signed by your Certificate Authority (CA) as an alternative to using certificates generated by AWS IoT

BYOC – Bring Your Own Certificate

CA = Certificate Authority

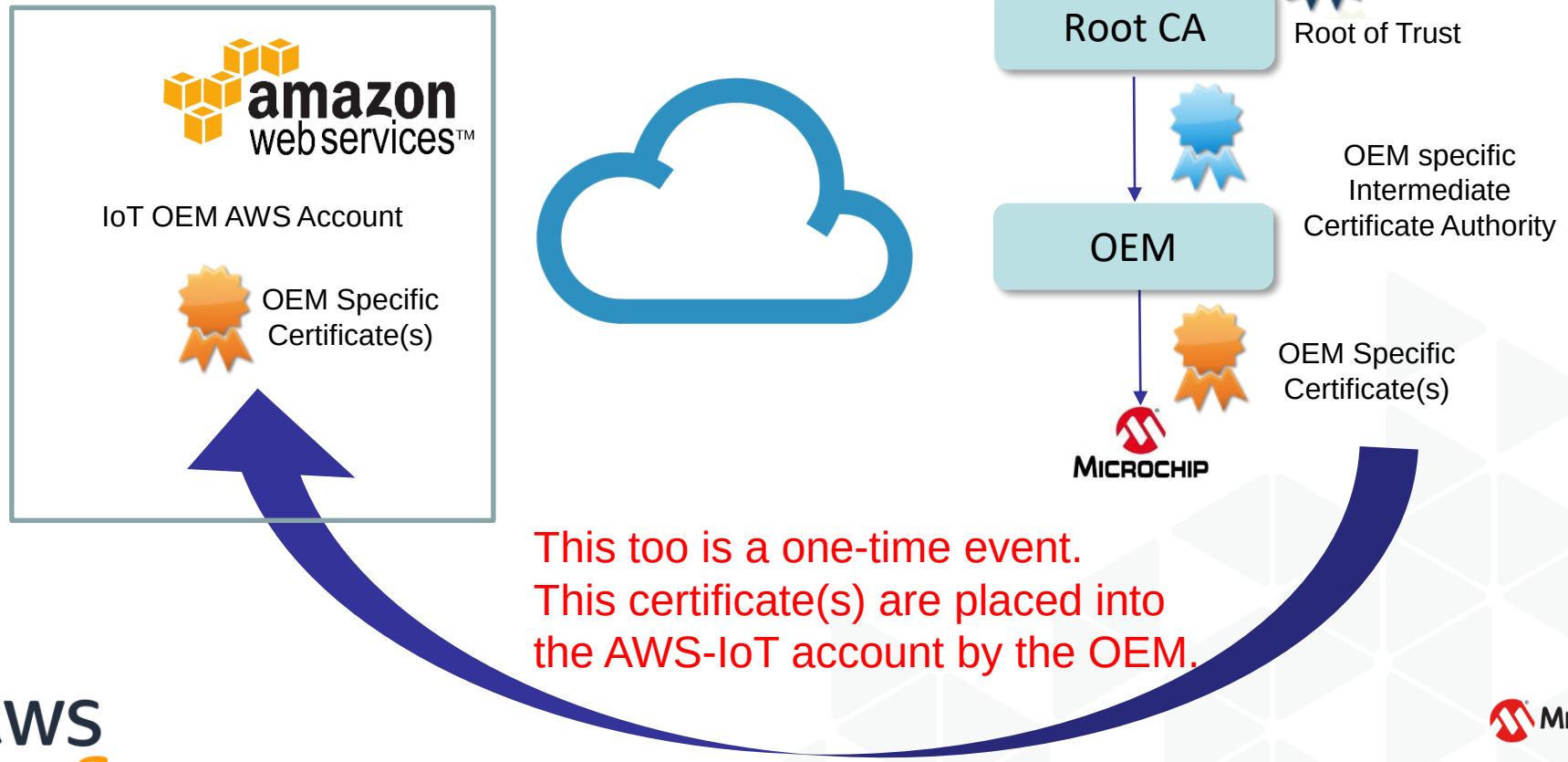




IoT OEM AWS Account

This is a one-time event.
The working certificate is needed to avoid the inconvenience and expense of having a CA provision everything.

BYOC – Bring Your Own Certificate



BYOC – Bring Your Own Certificate

SMART | CONNECTED | SECURE

CA = Certificate Authority

Root CA

Root of Trust

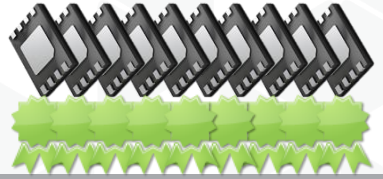
OEM

OEM specific Intermediate Certificate Authority

OEM Specific Certificate(s)

MICROCHIP

Massively Parallel Production:
Generate and Sign individual Device Certificates



amazon web services™

IoT OEM AWS Account

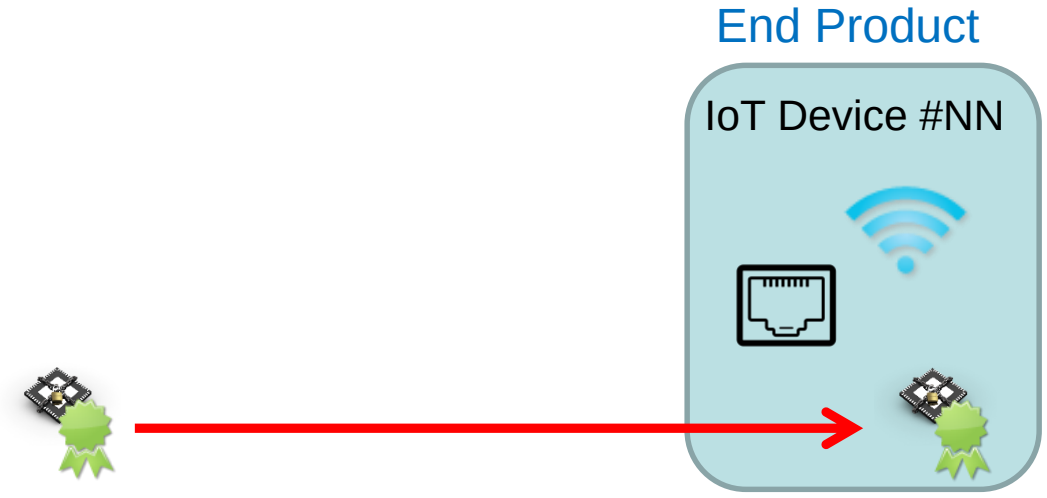
OEM Specific Certificate(s)

These provisioned chips are placed inside end product

aws

MICROCHIP

Provisioned Chip is soldered onto PCB



This is all that is needed to provision the end product

- Each end product no longer needs to be individually provisioned

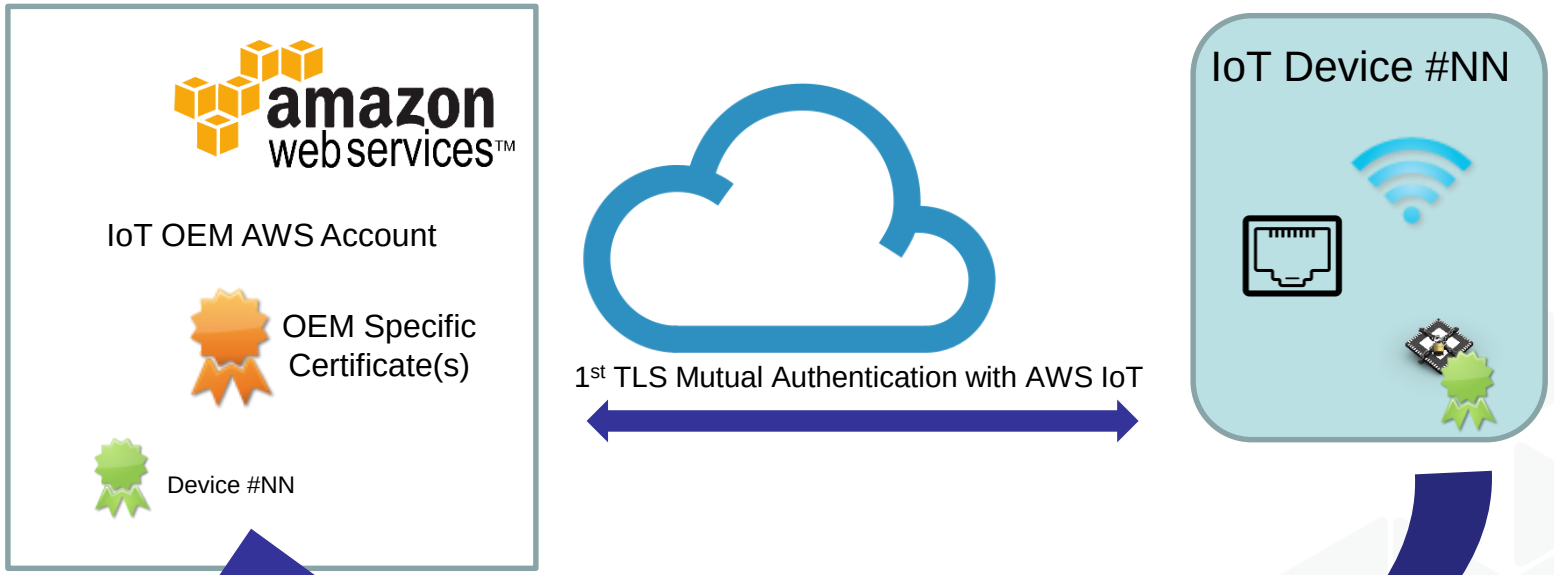
End product has some sort of internet connectivity

- WiFi, Ethernet, 802.15.4, etc…

JITR – Just In Time Registration

SMART | CONNECTED | SECURE

End Product



During first connection AWS IoT *automatically registers* the product in the correct account

How security is enhanced for …

Google Cloud Platform

SMART | CONNECTED | SECURE

MICROCHIP

Google requires individual registrations

Device identities are individually registered with GCP

- The connection to the Google servers may require TLS, but TLS is not the basis for on-boarding authentication for the GCP service
- Authentication done by use of a Token (JWT) and ECDSA

Once unique device identities are established, the users will place them in their GCP account(s)

- This will be done for every device they create
- This is done in the relative security of being signed into their own account

GCP allows use of your own certificate

SMART | CONNECTED | SECURE

CA = Certificate Authority

Root CA

Root of Trust

OEM

OEM specific Intermediate Certificate Authority

OEM Specific Certificate(s)

MICROCHIP

Google Cloud IoT Core
Device Management

Device Public Key

Inside each Individual Device Cert is the Device Public Key

The identities of each device are copied into the GCP account

Massively Parallel Production:
Generate and Sign individual Device Certificates

Devices

MICROCHIP

GCP Summary

Light JWT code

- Enables security for very small microcontrollers
- Capable of being used with 8bit MCUs

Agnostic of the TLS stack and MCU

- Design portability and flexibility

Every device identity has to be copied into GCP in advance.

How security is enhanced for …

SMART | CONNECTED | SECURE

MICROCHIP

A well defined ”Robust IoT Framework that is based on the DICE architecture from TPM.

Requires more processing performance than the other two but also offers a great deal of flexibilty and features

SMART | CONNECTED | SECURE

MICROCHIP

Secure device provisioning and authentication

Azure authenticates devices by the following two methods:

1. By unique identity key (security token) for each device, which can be used by the device to communicate with the IoT Hub.
   - The security token method provides authentication for each call made by the device to IoT Hub by associating the symmetric key to each call
2. By using PKI as a means to authenticate the device to the IoT Hub
   - Certificate-based authenticates a device at the physical layer as part of the TLS connection establishment.

The security-token-based method is considered less secure.

SMART | CONNECTED | SECURE

Similar to GCP, Azure requires individual registrations

Device identities are registered with Azure

- Azure servers use this identity to authenticate connection

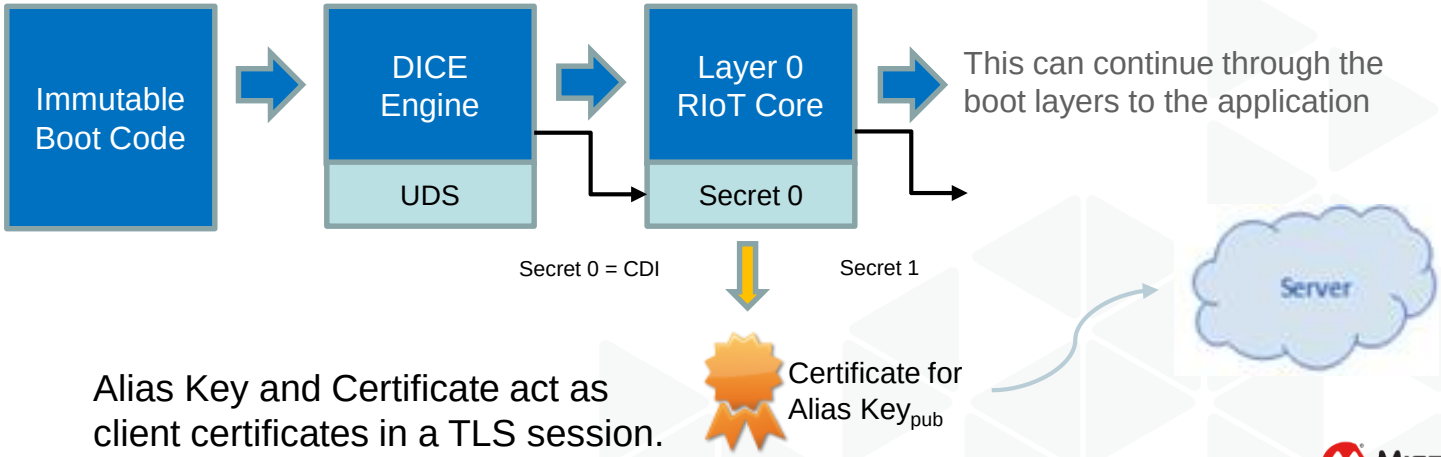
Once unique device identities are established, the account owners will place them in their Azure account(s)

- This will be done for every device they create
  - Loaded individually or as a group
- This is done in the relative security of being signed into their account

Device identity is pulled from the DICE / RIoT chain

The Device ID can be signed in one of two ways:

- Signed with DeviceID private key (self-signed)
  - Each device registers with the server (individual enrollment)
- Signed with manufacturer key, Vendor CA (vendor-certified)
  - This key can be registered for a group of devices (group enrollment)



Alias Key and Certificate act as client certificates in a TLS session.

Which Microchip Devices Support these methods?

Microchip Devices to Use

Support is always increasing, so watch this space!

Device(s)	TLS	Amazon AWS	Microsoft Azure	Google Cloud Platform
CEC1702	✓	✓	✓	
32-bit MCUs	✓			
32-bit MPUs	✓	✓	✓	
ECCx08	✓	✓		✓

Anything not listed here can be addressed by Custom Provisioning!

Microchip Devices to use

Devices from varied groups support these secure connections

- CEC1x02 devices
  - Code examples exist for AWS and Azure
- 32-bit MCUs - Cortex M23 and M33 based devices
  - New products with secure environment and HW Crypto blocks.
  - Code examples in development..
- 32-bit MPUs – Cortex A5 based devices
  - Code examples exist for AWS
- ATECCx08A devices
  - Code examples exist for AWS and Google Cloud Platform

TLS

aws

Google Cloud Platform

MICROCHIP