Protect Your Cloud Onboarding with the Latest LPC54S0xx Microcontroller

Donnie Garcia

MICR Systems & Applications Engineering

June 2019 | Session #AMF-SOL-T3647

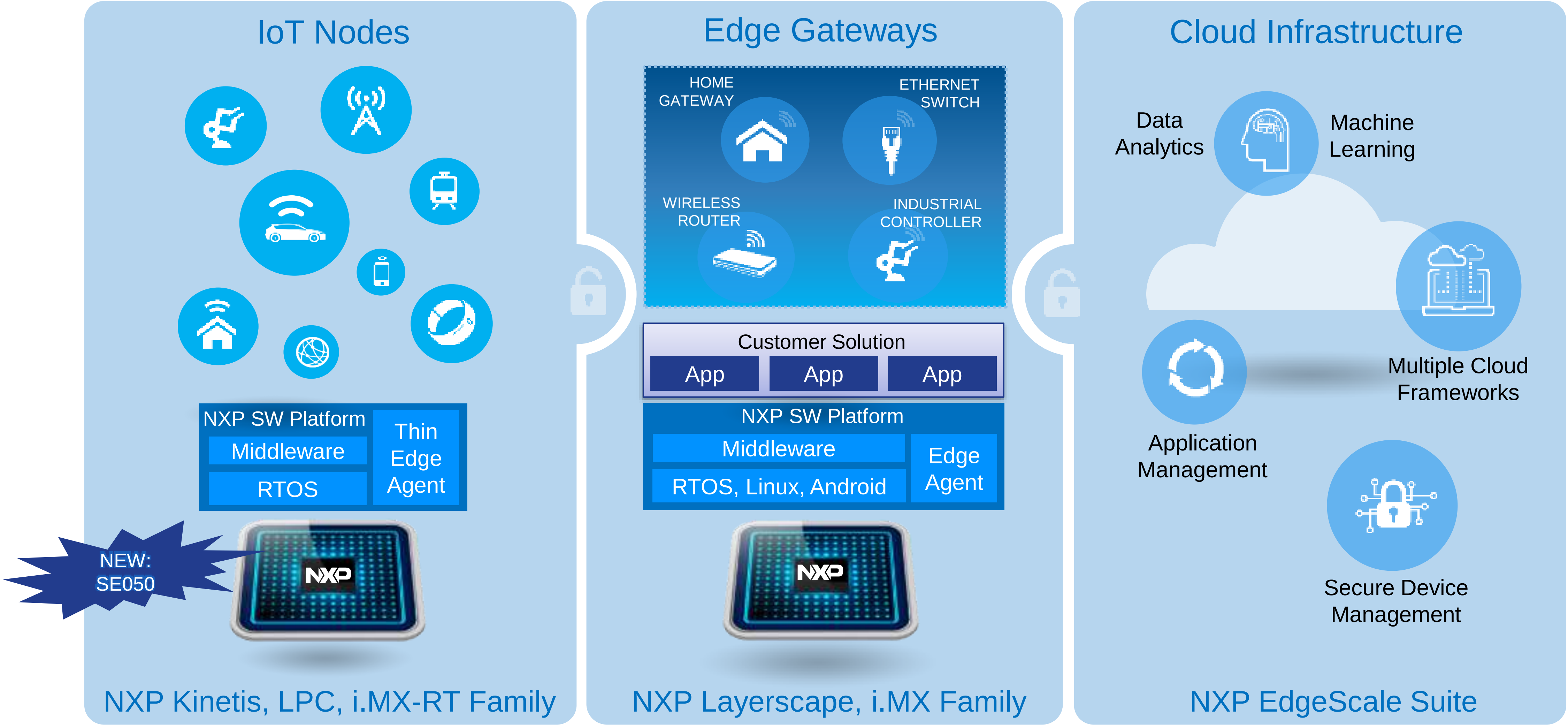
SECURE CONNECTIONS FOR A SMARTER WORLD

Company Public – NXP, the NXP logo, and NXP secure connections for a smarter world are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2019 NXP B.V.

Agenda

- Relevance of IoT Security
- IoT Security: Threats, Principles and Lifetime
- AWS at a Glance
- LPC54S0xx: Protecting Cloud On-boarding
- Q & A

NXP Solutions for Edge Computing



Relevance of IoT Security

Consumer IoT Device Attack Trends

Attack method	Profitability	Comment	Trend
DDoS attacks	+	Still growing in size - simple	↑
Spam attacks	– –	Not the easiest way to spam	↓
Cryptocurrency mining	●	Depends on the coin price	→
Ransomware/locker	+	Might work on some devices	↑
Blackmail/extortion	●	Does not scale well – depends	→
Pranks/nuisance	– –	Not done by cyber criminals	→
Information stealing	●	Done because it's simple	↑
Click fraud	+	Often overlooked - profitable	↑
Premium services	+	Difficult to conduct	↓
Sniffing network traffic	●	Difficult with SSL/TLS	↓
Pivoting/attacking LAN	+	Infecting attached computers	↑
Proxy	●	Not very lucrative, but useful	→

https://www.rsaconference.com/writable/presentations/file_upload/sem-m03d-profiting-from-hacked-iot-devices-coin-mining-ransomware-something-else.pdf

- Profitability motivates the IoT attacker
- DDoS attacks are enabled by dark web store fronts
- As the value of devices and the data they handle increases, Ransomware or device lock out attacks will rise
- IoT devices with weak cybersecurity allow attackers entry into protected networks

Legislation

https://www.youtube.com/watch?v=YxC1kcZDMyc&feature=youtu.be

- Convergence on IoT security guidelines from many angles (foreign and domestic)
- OWASP (Open Web Application Security Project has a nice list

Threats, Principles and Lifetime

Attacks Occur in Many Forms: Different Types and Locations

Attack type

- Physical – making use of physical properties or deficiencies in the device
- Logical – by sending malicious messages, the software will misbehave

Adversary's location

- Local – adversary must be in the proximity of the device
- Remote – adversary can be anywhere

IoT Security Threats and General Protection Principles

Threat Spectrum

	Physical	Logical
Local	• Power analysis • Light attacks • Glitching	• All local interfaces: – Exploiting JTAG – Serial – USB – …
Remote	• Rowhammer • Cache Timing	• Buffer overflow Heartbleed • Flooding/DoS

Solution Rationale Against Threats

3 Physical	2 Logical
If an attacker can get local access to the device, make a cost/benefit trade-off and protect against relevant local physical attacks over the lifetime of the device	If an attacker can get local access to the device, aim to protect against local logical attacks. Reason: can be automated and executed by laymen

1 Aim to protect against remote attacks.
Reason: scalable attacks can be automated and executed by laymen from anywhere in the world

Level of importance to ensure security against threats *High* *Higher* *Highest*

A Connected/smart Device is Vulnerable Throughout its Lifecycle

Product lifecycle

Procure, Develop, Manufacture, Distribute

Local Attacks (physical and logical)

- Extract keys/certificates
- Overproduction of original device
- False certificate/private key injection
- Malicious image loading
- Ccounterfeits of devices
- IP Theft

Remote attacks could be made against infrastructure

Onboard, Operate and Update

Local Attacks (physical or logical) – Per Device

- Tamper the IC to obtain access to data and SW and re-use for remote attacks (Trojan horse, DoS on Cloud, …)
- Especially dangerous for non-diversified Symmetric key protection: “Break one, Break all”

Remote attacks – Scalable to all devices

- Create unauthorized connection to extract data, abuse functionality or inject malware to turn device into a bot
- Perform malicious software update to do the same

Decommission

Local Attacks (physical and logical)

- Extract credentials (user data, keys, certificates)
- Inject malware to network

Remote attacks are possible by re-commission the device to attack the network or cloud

Derived Features for IoT Devices

Secure system lifecycle	System is able to securely go through its different life stages, including: Power-up / Boot phase, debug, OTA updates of FW and SW and decommissioning
Crypto & Key protection	A device must provide means for protected secure root key provisioning and storage of key and certificate credentials, and handle the security of derived keys.
Resource isolation (HW and SW)	Minimize the attack surface by isolating HW (like memory and processing) and SW resources used for platform security features from other parts of the IoT system
Run time integrity and attestation	The run-time integrity of a system is ensured and can be (remotely) validated – This validation is (remote) attestation

LPC54S0xx Security Technology

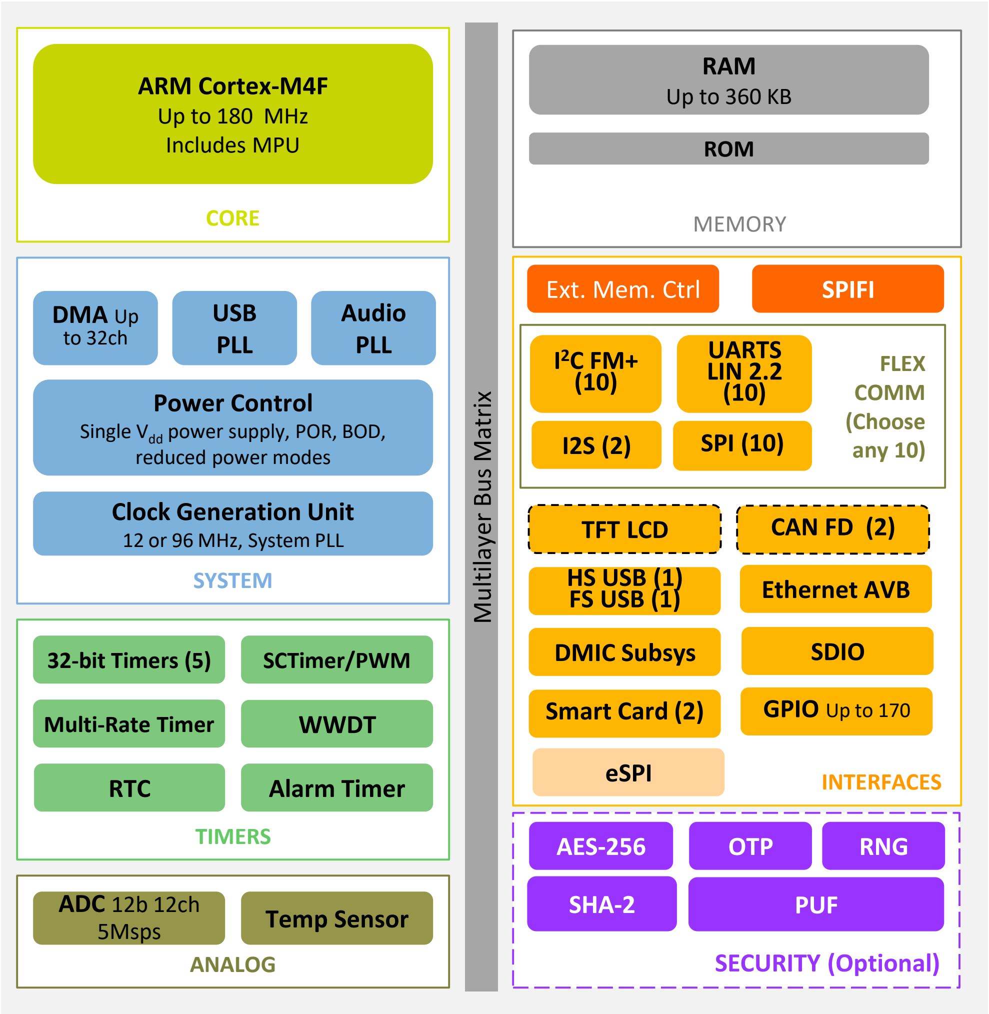
LPC54Sxx Block Diagram

High performance with high-end Graphical User Interface and security

- Cortex-M4F, 180MHz
  - Up to 360 KB RAM
  - 16KB EEPROM
  - XIP from QSPI via SPIFI
  - External Memory Ctrl (up to 32 bits)

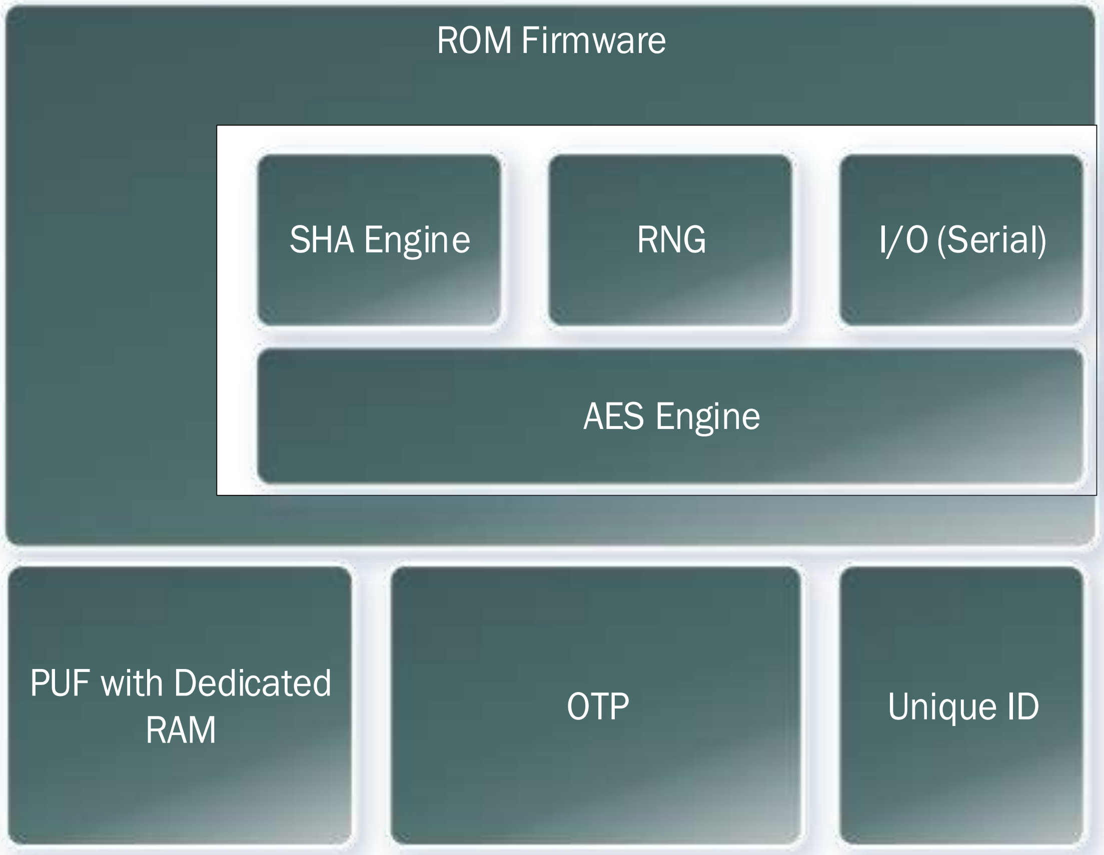
Key Features

- Graphic LCD with resolutions up to 1024 x 768
- CAN-FD controller x2 (LPC54608)
- eSPI interface (slave and LPC bus device functionality)
- Digital mic subsystem supporting voice detection
- Hi-Speed and Full Speed USB
  - USB: 1x HS (H/D) w/on-chip HS PHY
  - XTAL-less FS USB Dev
- FlexComm: flexible serial connectivity
- Advanced Security Option:
  - AES-256, SHA-2, True RNG
  - PUF for key storage
  - HW diversified OTP Key Storage
  - Secure boot using 2048-bit RSA authentication and SHA-2 verification
  - Encrypted boot using AES-GCM mode



LPC54Sxx Security Sub-system

- ROM supporting secure boot methods
  - Authentication, Encryption, combination options
- AES Engine
  - Supports 128, 192 or 256 bit keys
  - Encryption modes: Electronic Codebook (**ECB**), Cipher Block Chaining (**CBC**), Cipher Feedback (**CFB**), Output Feedback (**OFB**), Counter (**CTR**), Galois Counter Mode (**GCM**)
- SHA Engine
  - Support SHA1 (160 bit) and SHA2 (256 bit)
- Physically Unclonable Function (PUF)
  - Device unique root key (256 bit strength)
  - Can store key sizes 64 bit to 4096 bit
  - Index 0 Keys routed to AES engine via direct HW bus
- 128-bit UUID per device
  - RFC4122 compliant
- Random Number Generator (RNG)
  - FIPS 140-1 compliant
- HW diversified OTP key
  - Key stored in OTP is scrambled using device unique ID

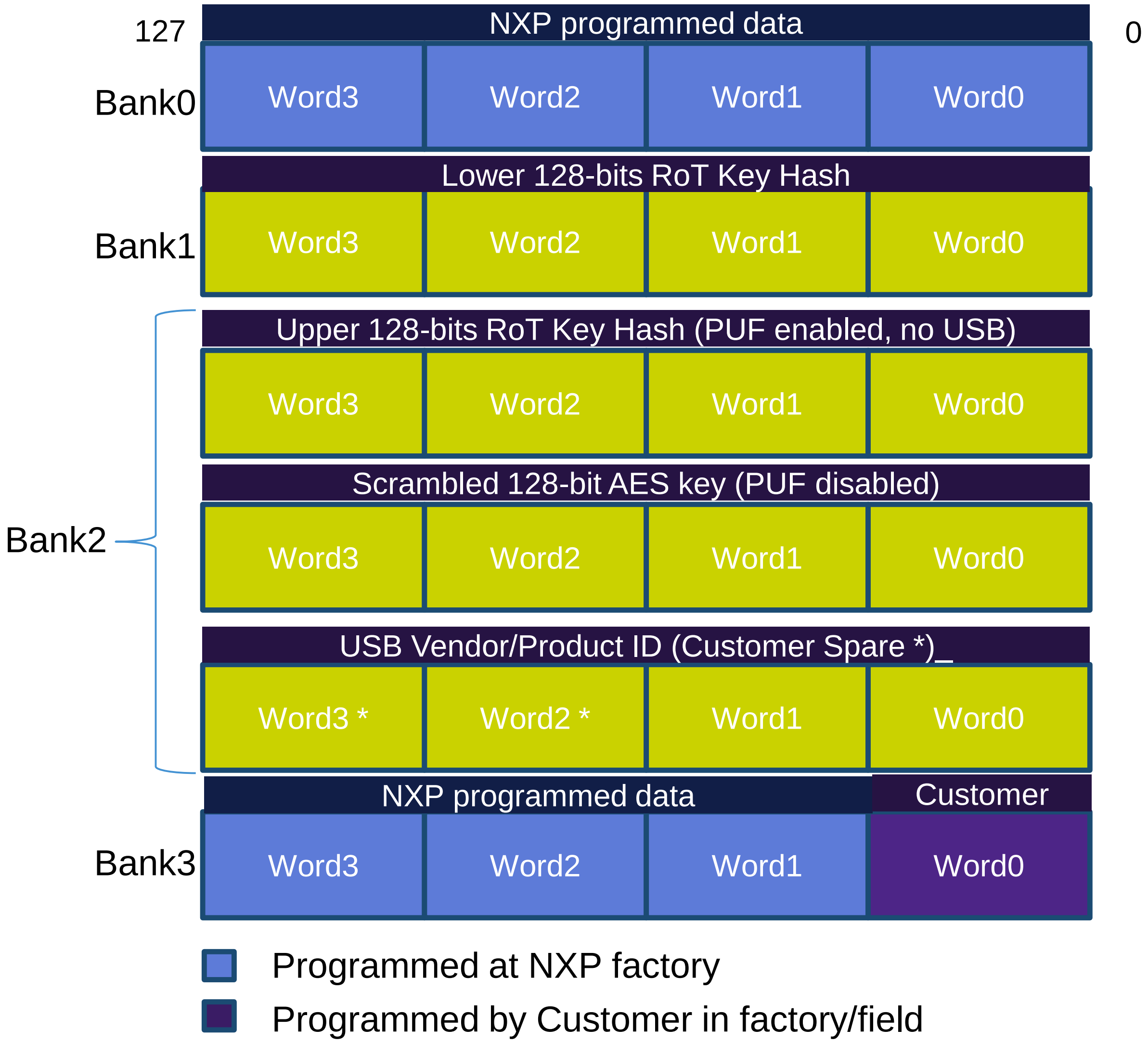


OTP

OTP Layout – LPC54S0xx

OTP provides tamper proof storage for key material and boot control

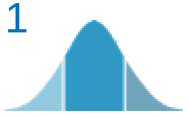
- OTP is organized as four 128-bit banks
- Each 32-bit word can be read/write lockable
- Bank 1 (either):
  - Lower 128 bits of RoT key hash
  - Customer data (encrypt only)
- Bank 2 (either):
  - Upper 128 bits of RoT key hash
  - Scrambled 128-bit AES key (PUF disabled)
  - USB vendor/product IDs
  - Customer data (encrypt only, PUF enabled)
- Bank 3 – word 0 of controls secure boot behavior along with key revocation list



PUF

SRAM PUF Technology

1



Process Variation

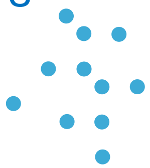
Naturally occurring variations in the attributes of transistors when chips are fabricated (length, width, thickness)

2

SRAM Start-up Values

Each time an SRAM block powers on the cells come up as either a 1 or a 0

3



Silicon Fingerprint

The start-up values create a 100% random and repeatable pattern that is unique to each chip

4

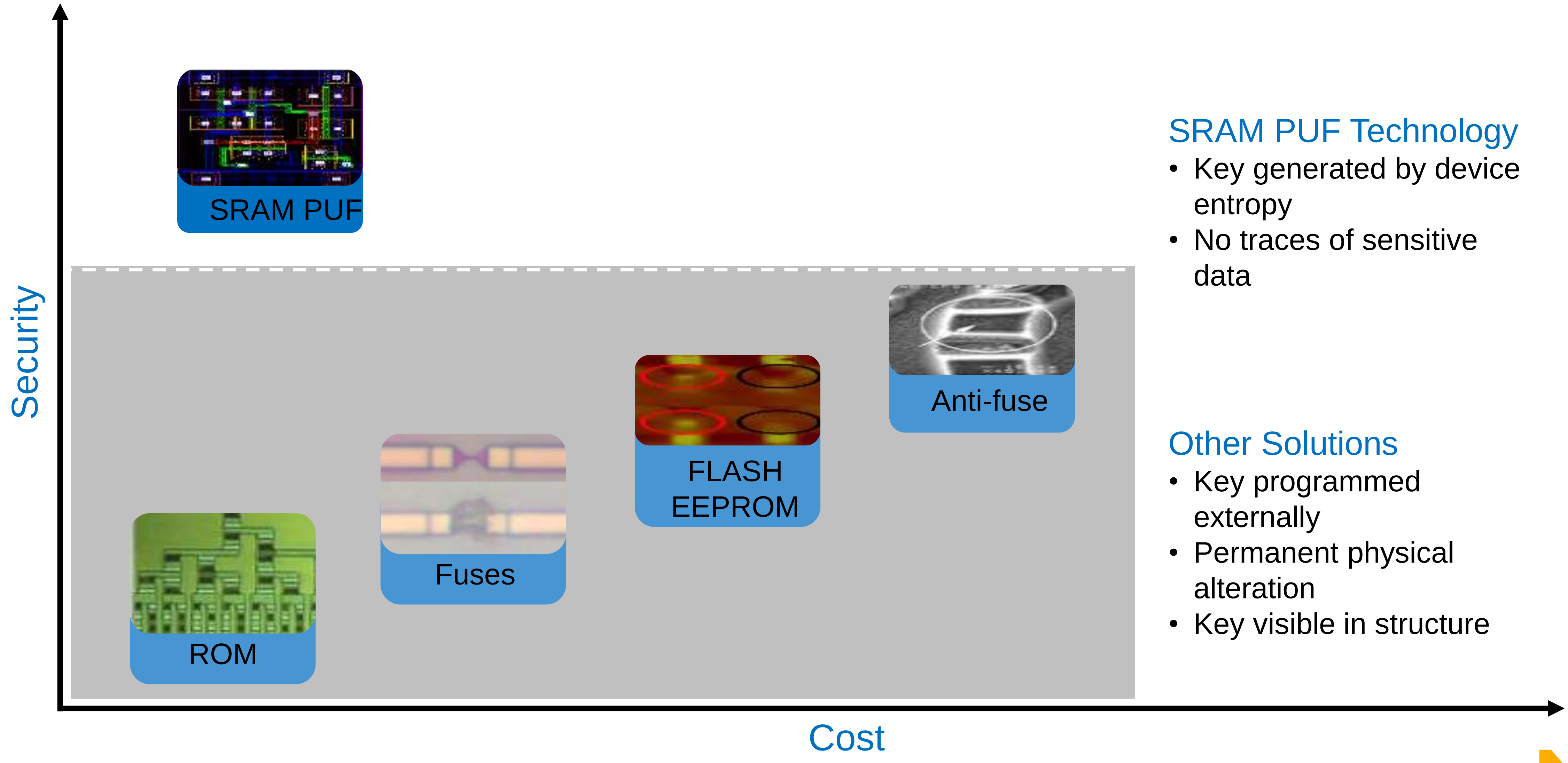
SRAM PUF Key (K_{PUF})

The silicon fingerprint is turned into a secret key that builds the foundation of a security subsystem

SRAM PUF Benefits

- Device-unique, non-reproducible fingerprint
- Leverages entropy of mfg. process
- No key material programmed

SRAM PUF Advantages



SRAM PUF Technology

- Key generated by device entropy
- No traces of sensitive data

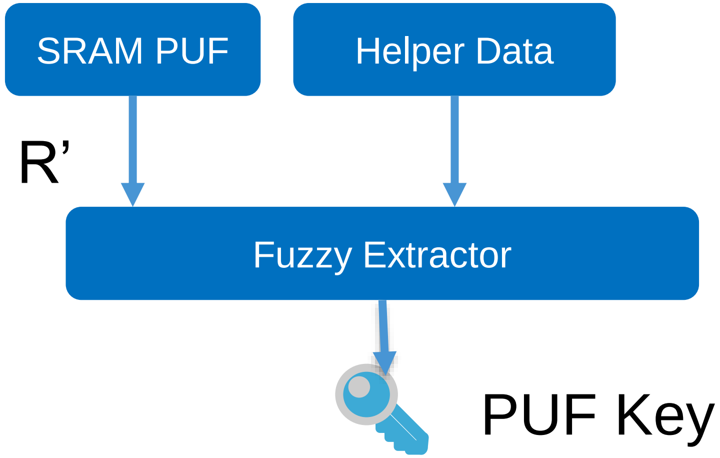
Other Solutions

- Key programmed externally
- Permanent physical alteration
- Key visible in structure

Key Provisioning Based on SRAM PUF

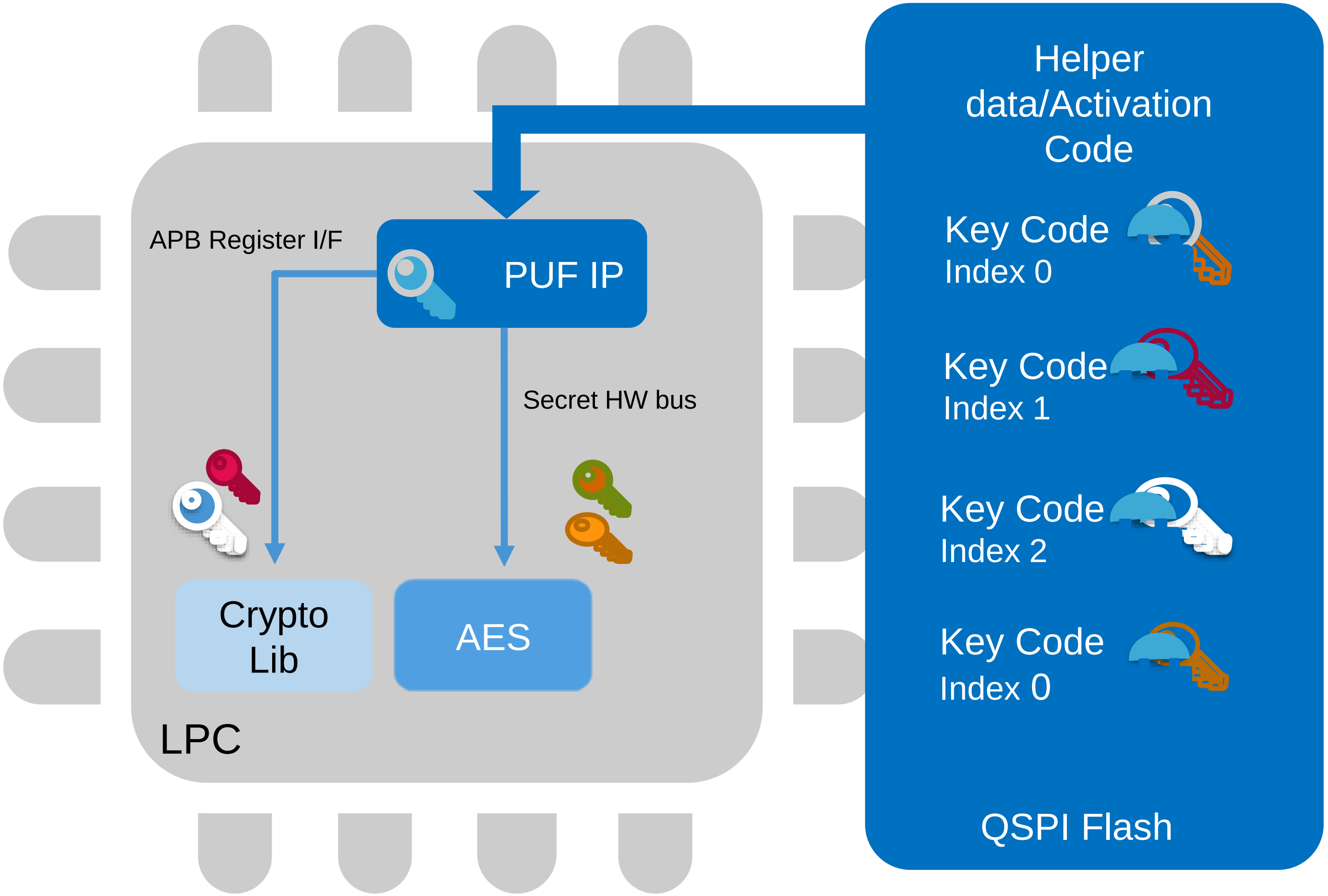
1. Enrollment Mode

2. Key Reconstruction Mode



- SRAM PUF response (R) is a noisy fingerprint of the chip
- PUF IP implements the Fuzzy Extractor or Helper Data Algorithm
  - Error correction
  - Privacy Amplification
- Two operation modes:
  - Enrollment mode
  - Key Reconstruction Mode

PUF Key Store



LPC PUF Features

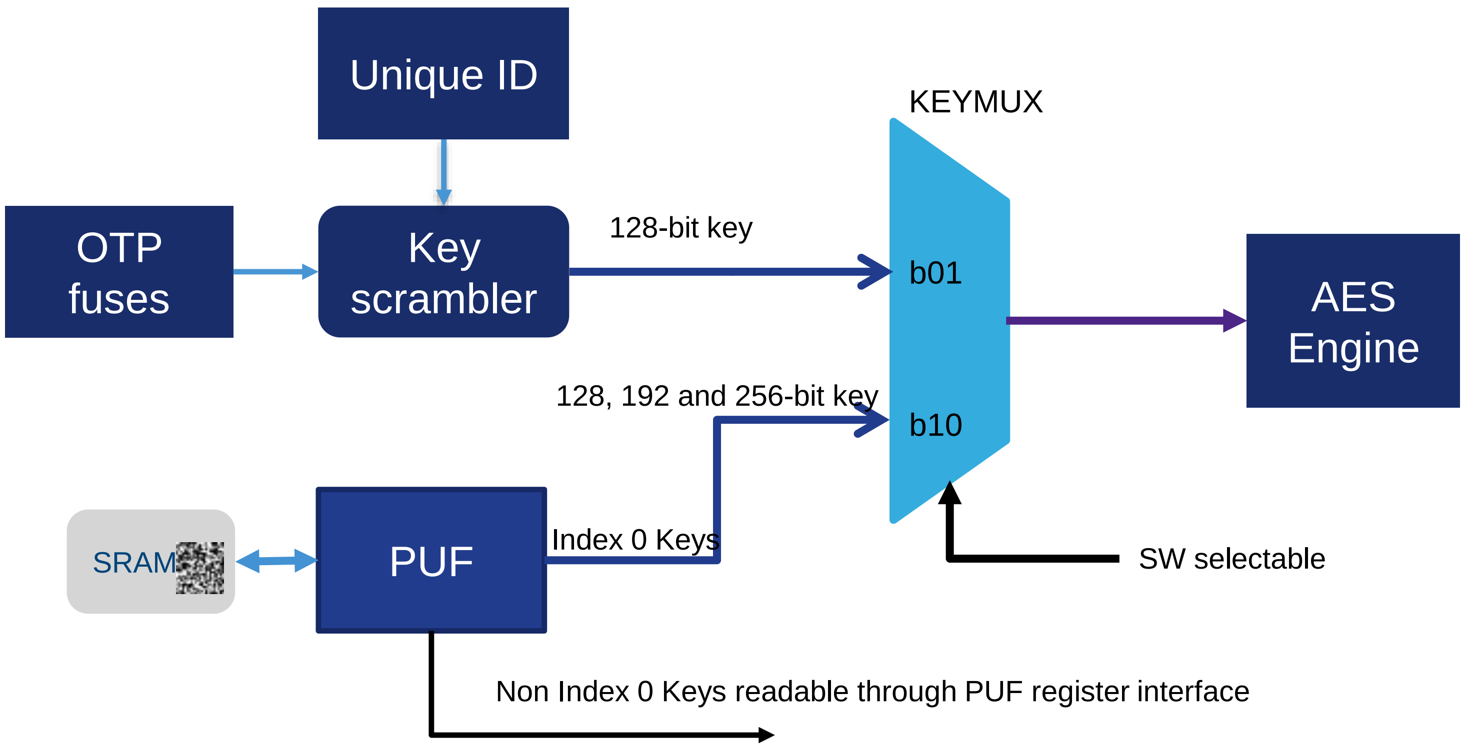
- 256 bit strength Root key
- Supports wrapping of keys
  - 64 to 4096 bits keys
  - Generation of Intrinsic keys (random key)
  - Index 0 accessible through HW secret bus
  - Other indexes through register I/F

Key Management

Key Management – HW AES Key Paths

- Critical keys feed directly to AES engine via HW bus
- No access to secret keys (Index 0) via SW readable registers
  - Except during provisioning

- PUF derives unique root key (KPUF) per device from SRAM fingerprint
  - Eliminates complexity of generating unique keys per device during provisioning
  - Protects credentials on a per device basis



Secure Boot ROM

LPC54Sxx Secure Boot ROM Features

- Support following secure boot mode
  - Authentication only image: Public Key signed image
  - Encrypted image: Symmetric key encrypted image with and MAC authentication
  - Enhanced secure image: Symmetric key encrypted image with Public Key authentication

- Support Public Keys & Image Revocation

- Support redundant boot image on external SPI flash

Secure Boot ROM – Option Details

- Secure Boot
  - Authentication only images:
    - RSA signature verification with public key (2048-bit modulus and 32-bit exponent)
    - Image key certificate support with revocation capability
    - Uses OTP to authenticate public key
      - SHA256 digest of public key should be stored in OTP

- Encrypted Boot
    - Uses AES-GCM mode to decrypt and authenticate image
      - 256-bit key using PUF

        OR

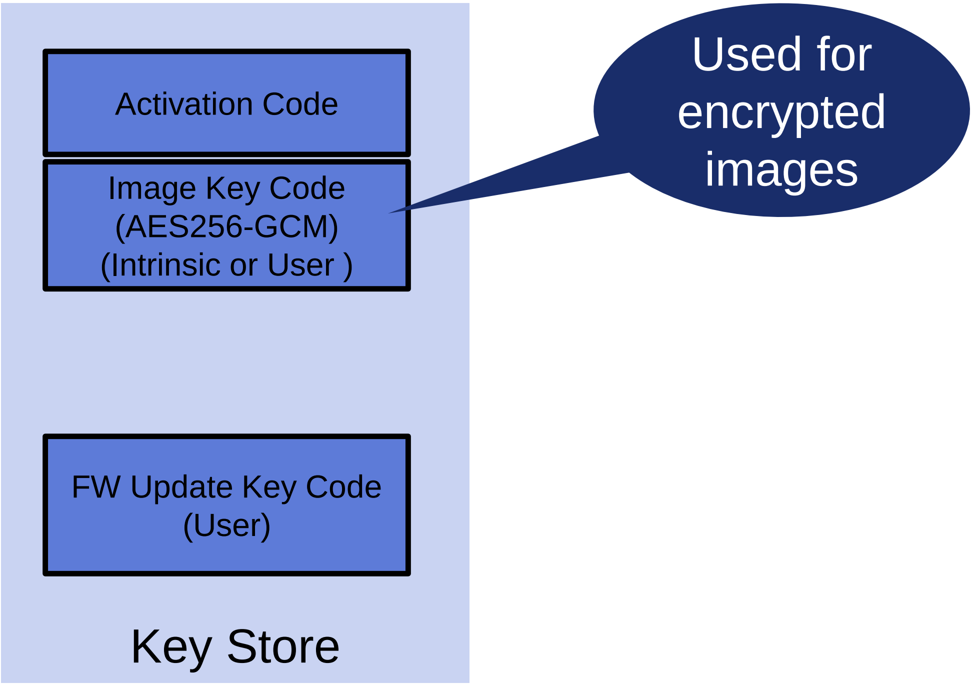
      - HW diversified 128-bit key using OTP

- Enhanced images support based on security policy

Key Store Details

- Activation code
  - ~1KB of data generated during PUF enrollment
  - Helper data (~Error Correction Code) to reconstruct root key
  - Generated during provisioning

- Key codes
  - User keys
    - Pre-shared keys
    - Provisioned during manufacturing
  - Intrinsic keys
    - Random keys



Key Management

Key Management Table

Authentication Only, No Encryption (See Slide 47)

Key Name	Description	Owner	Key Generation	Key Use	Key Storage
Root of Trust Private Key	RSA 2048 Private key used for creating signatures of the image key certificate	OEM -OR- CA	Generated by the image creation tool (python script). Use of key material can be password protected in this tool	Used to sign the image key certificate which is part of image data	Use a certificate authority -OR- Trusted OEM machine must have OpenSSL and should have a strong RNG and be "Air Gapped"
Root of trust Public Key	Associated RSA 2048 public key for authenticating boot code. This key is inserted into the image certificate which becomes part of the boot data	OEM	Tool can generate HEX output of the hash of the public key to be stored in OTP which is checked during booting	Used to validate the image certificate which is holds the Image Public key. Not a secret key, checked for integrity by Root of Trust Hash.	Part of the boot image Hash stored on chip OTP for integrity check
Image Private Key	RSA 2048 Private key used for creating signatures of application binaries	OEM	Generated by the image creation tool (python script). Use of key material can be password protected in this tool		Trusted OEM machine must have OpenSSL and should have a strong RNG and be "Air Gapped"
Image Public Key	Associated RSA 2048 public key for authenticating boot image. This key is inserted into the certificate which becomes part of the boot data	OEM	Generated by the image creation tool (python script)	Used to validate the boot image upon every reset	Part of the boot image

Key Management Table

Signed and Encrypted (Enhanced with PUF); See slide 49 for Fuse settings and previous slide

Same keys as detailed in previous slide, but also, the below

Key Name	Description	Owner	Key Generation	Key Use	Key Storage
PUF Boot Encryption Key (Image Key)	AES256bit symmetric key which is used to encrypt application code and data.	OEM	Tool generates AES key LPC Chip Set PUF KEY encrypts this key	Decrypt the signed image (GCM)	Built during manufacturing time. The plain text key is given to PUF and encrypted for in system storage in external flash.
K_{PUF} Hardware unique key	AES256 bit symmetric key which is used to protect the PUF Boot Encryption Key.	LPC Chip	Generated by the chip itself (PUF SRAM Fingerprint based)	Used to create other keys that go in the key store (like the PUF Boot encryption key)	Intrinsic to PUF
Activation Code	Not a Key, but data needed to support the PUF	LPC Chip (external Flash)	Generated during the enrollment phase of PUF	Not a key, but helper data needed to support PUF	External Flash (see slide 64)

Counterfeit protections

Multiple options for establishing unique identity for the Chip

1) RFC4122 compliant Unique ID

2) OTP Key diversified by chip Unique ID can be used to create encrypted private key material

3) PUF generated key can operate in a similar capacity

Keys are protected by dedicated interface to HW AES engine

Onboarding

Chip specific unique and protected keys along with secure boot flow protect OEM installed cloud credentials

Cloud credentials become part of the secure boot image that is protected for integrity and confidentiality

During manufacturing cloud credentials are encrypted with chip specific unique & protected keys

System Integrity

Secure boot functions provide the foundation for establishing trust in the device functions

ROM provides an immutable secure boot flow to support recovery from system run away scenarios

Arm MPU for memory partitioning for logical security

Secure Communication

TLS Stacks (Arm Mbed TLS) use hardware managed keys

Option for AES engine to use OTP or PUF generated keys

Hardware acceleration for AES and SHA-2 (SHA-256)

Data Confidentiality

Based on **device policies**, data stored in system is protected by hardware managed keys

Option for AES engine to use OTP or PUF generated keys

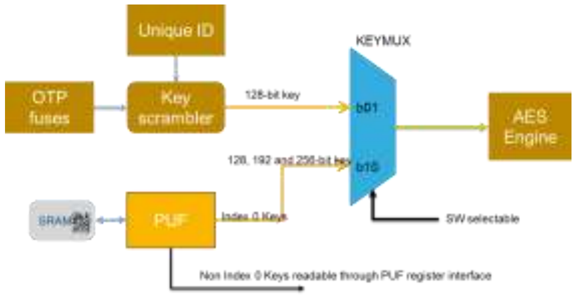
Hardware acceleration for AES and SHA-2 (SHA-256

Secure firmware update

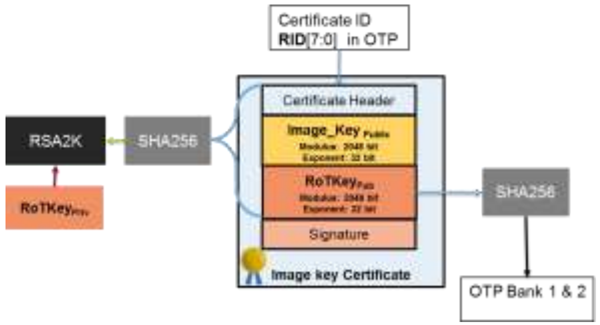
New firmware applied to the system must pass the secure boot flow

ROM support for up to 8 revocations

Key Management

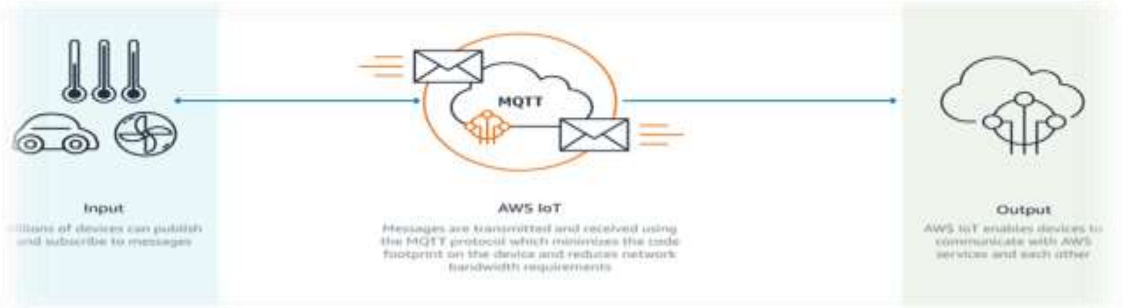


Secure Boot



AWS IoT at a Glance

Amazon Web Services and AWS IoT



All AWS Services

Compute
- EC2
- Lightsail
- Elastic Container Service
- EKS
- Lambda
- Batch
- Elastic Beanstalk

Storage
- S3
- EFS
- Glacier
- Storage Gateway

Database
- RDS
- DynamoDB
- ElastiCache
- Neptune
- Amazon Redshift

Management Tools
- CloudWatch
- AWS Auto Scaling
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Systems Manager
- Trusted Advisor
- Managed Services

Media Services
- Elastic Transcoder
- Kinesis Video Streams
- MediaConvert
- MediaLive
- MediaPackage
- MediaStore
- MediaTailor

Mobile Services
- Mobile Hub
- AWS AppSync
- Device Farm
- Mobile Analytics

AR & VR
- Amazon Sumerian

Application Integration
- Step Functions
- Amazon MQ
- Simple Notification Service
- Simple Queue Service
- SWF

Customer Engagement
- Amazon Connect
- Pinpoint
- Simple Email Service

Migration
- AWS Migration Hub
- Application Discovery Service
- Database Migration Service
- Server Migration Service
- Snowball

Networking & Content Delivery
- VPC
- CloudFront
- Route 53
- API Gateway
- Direct Connect

Developer Tools
- CodeStar
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline
- Cloud9
- X-Ray

Machine Learning
- Amazon SageMaker
- Amazon Comprehend
- AWS DeepLens
- Amazon Lex
- Machine Learning
- Amazon Polly
- Rekognition
- Amazon Transcribe
- Amazon Translate

Analytics
- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- QuickSight
- Data Pipeline
- AWS Glue

Security, Identity & Compliance
- IAM
- Cognito
- Secrets Manager
- GuardDuty
- Inspector
- Amazon Macie
- AWS Single Sign-On
- Certificate Manager
- CloudHSM
- Directory Service
- WAF & Shield
- Artifact

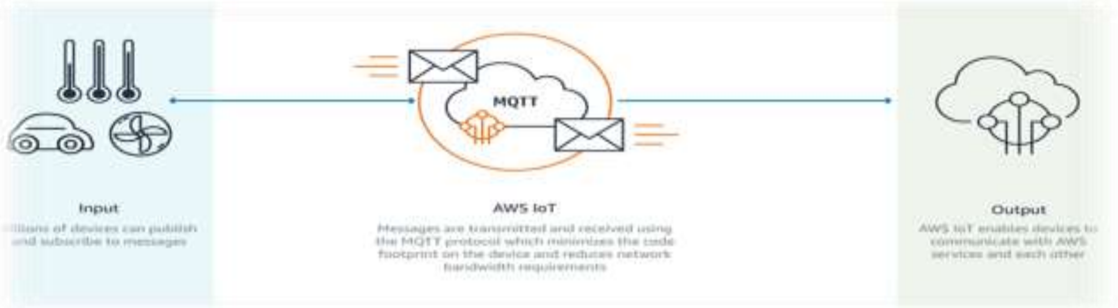
Business Productivity
- Alexa for Business
- Amazon Chime
- WorkDocs
- WorkMail

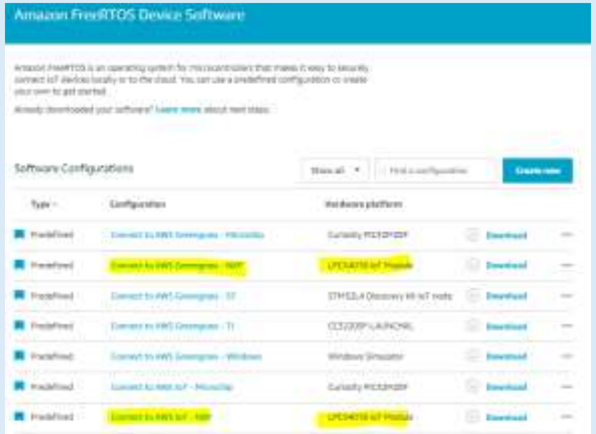
Desktop & App Streaming
- WorkSpaces
- AppStream 2.0

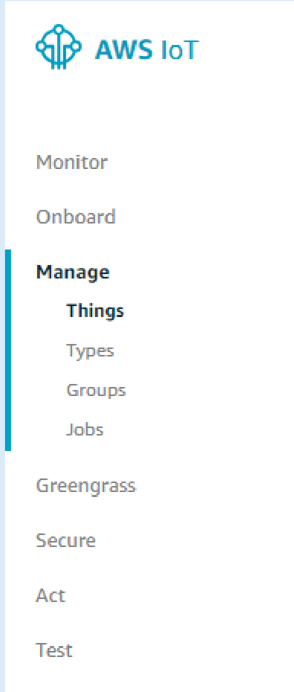
Internet of Things
- IoT Core
- IoT 1-Click
- IoT Device Management
- IoT Analytics
- Greengrass
- Amazon FreeRTOS

Game Development
- Amazon GameLift

AWS IoT Device and Cloud Views

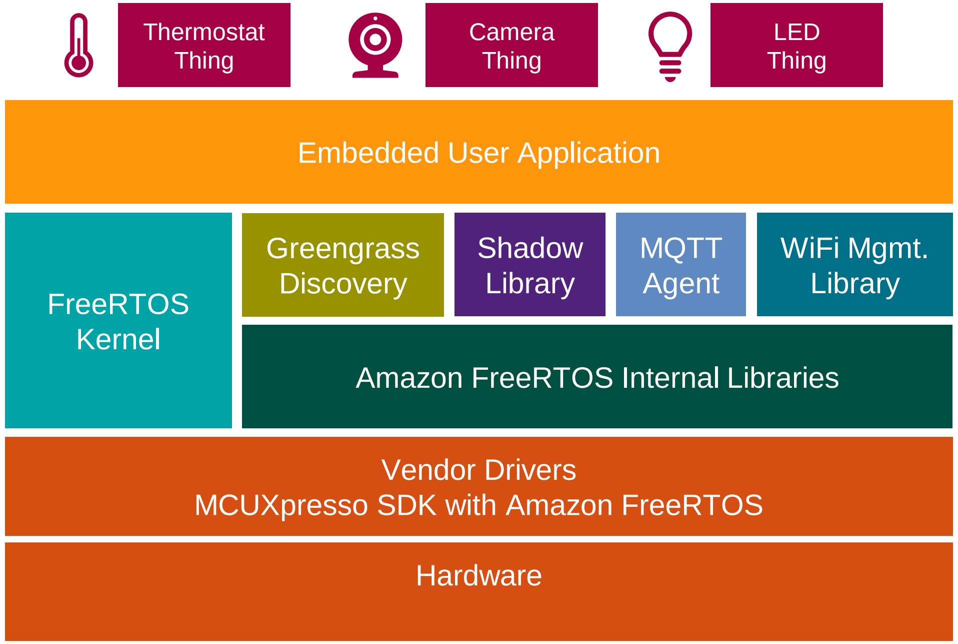






Amazon FreeRTOS at the Device

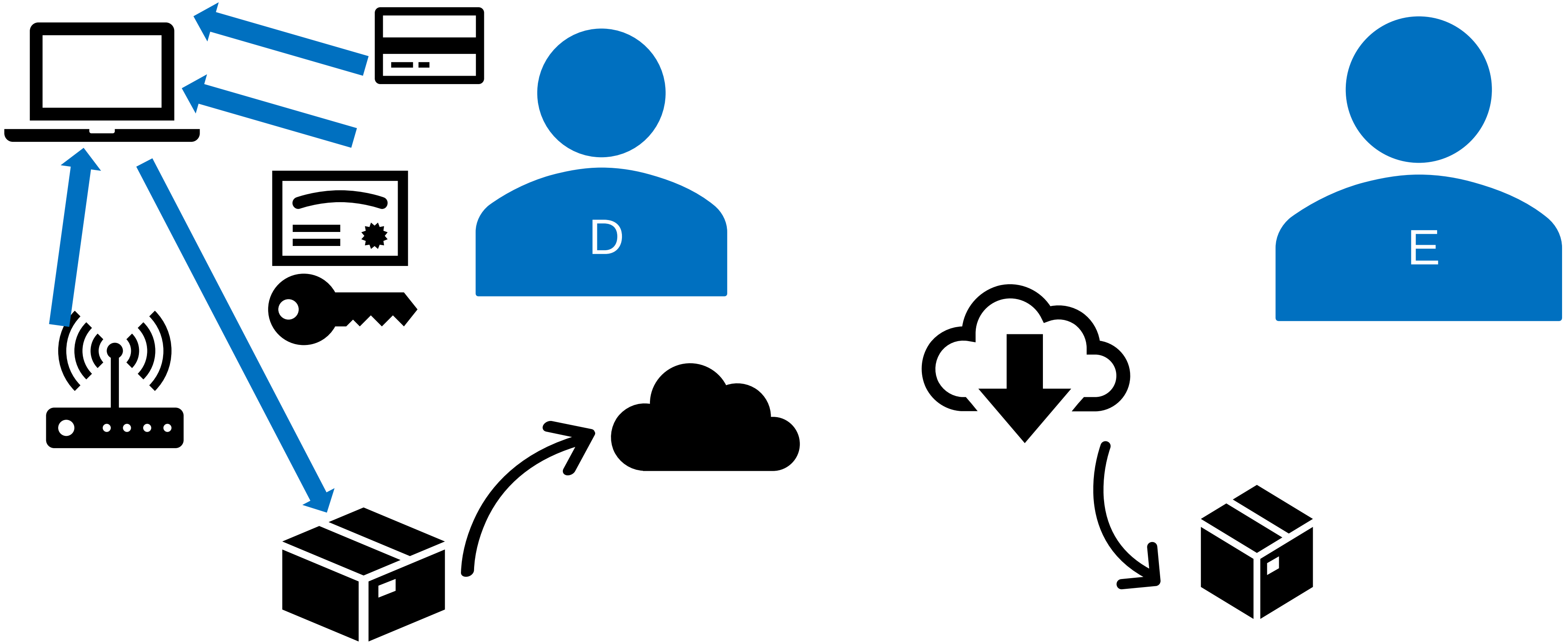
The FreeRTOS kernel is now an AWS open source project, and these pages are being updated accordingly. AWS are pleased to announce immediate availability of the MIT licensed Amazon FreeRTOS operating system, built on the FreeRTOS kernel v10.



AWS IoT: What Can Go Wrong?

Actual Events, Names and Faces Have Been Changed…

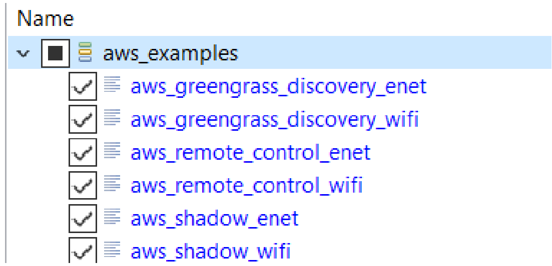
- Developer D from one Company is working jointly with Developer E from a different Company on benchmarking performance for a new processor and memory architecture
- They decide together to use an Amazon FreeRTOS example as a “typical IoT application”
- Developer D sets up an AWS account and gets the “MCUXpresso SDK Remote Control” application working and enables Show-Run-Time stats for Amazon FreeRTOS
- Excited about the results and working towards a deadline Developer D shares his work with Developer E.



Developer D: Uses personal credit card to create AWS account, creates device credentials for App, sets default policies for the device and the Smart phone App that controls it, uses home WiFi Credentials to get the app working then post the package for Developer E

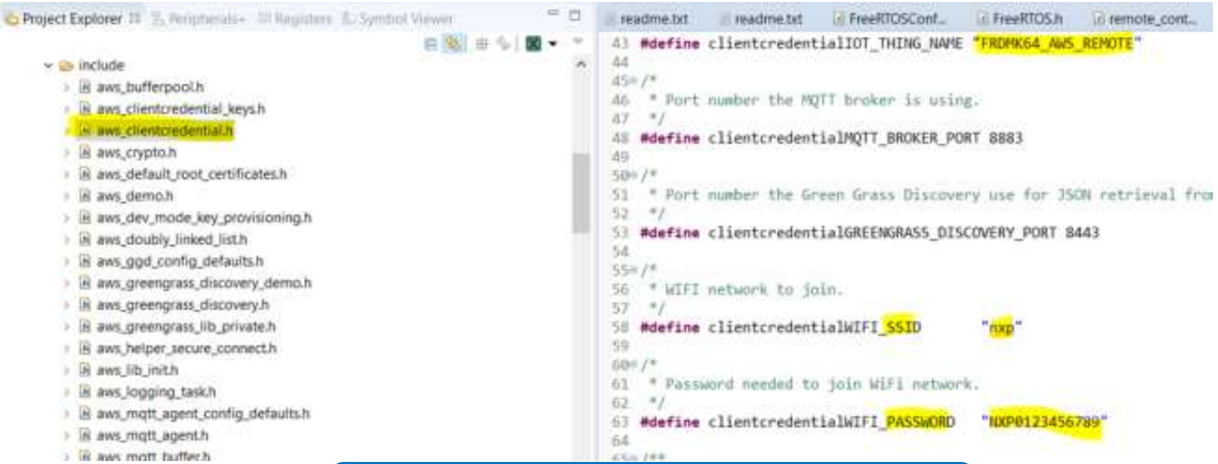
Developer E: Now has access to Developer D's Wifi SSID and password. With the device credentials Developer E can make a counterfeit device and use it to push large amounts of data to Developer D's AWS account leading to data fees charged to personal Credit Card of Developer D

Amazon FreeRTOS Examples are Part of MCXpresso SDK

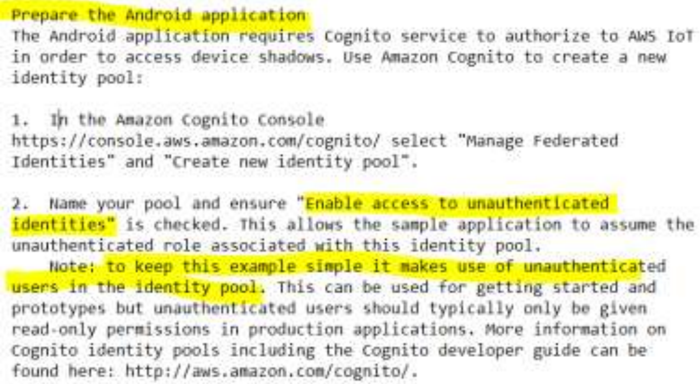


Great Examples!

Device private key stored as plain text in a header file



Device Name, WiFi Credentials are plain text



Unauthenticated entities are allowed

AWS Vulnerabilities: What Needs to be Protected

AWS Security Goals Statement

https://docs.aws.amazon.com/iot/latest/developerguide/iot-security-identity.html

Security and Identity

AWS IoT Authentication

- You are responsible for managing device credentials (X.509 certificates, AWS credentials) on your devices and policies in AWS IoT. You are responsible for assigning unique identities to each device and managing the permissions for a device or group of devices.

Version 1 updated May 28, 2018 5:59:39 PM -0500


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Connect",
        "iot:Receive"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

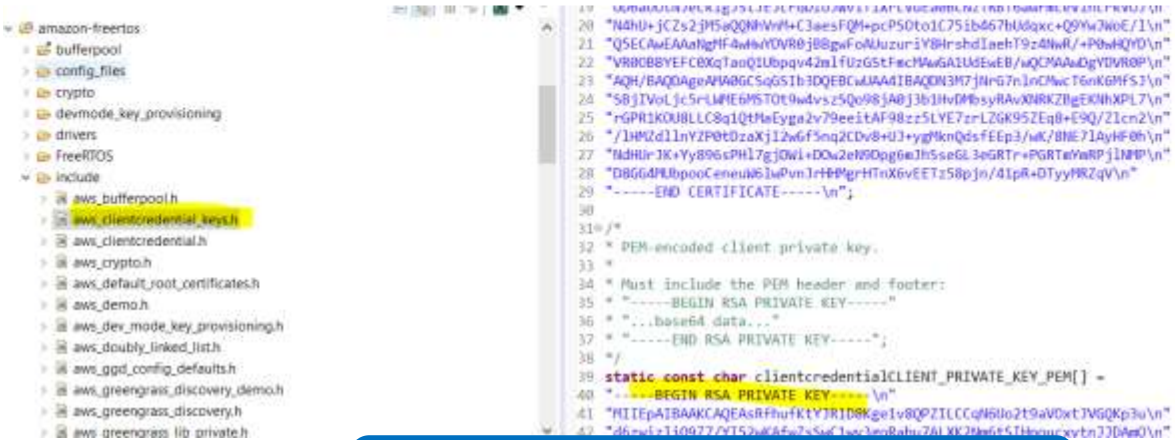
Other Application Specific Needs

- WiFi Credentials
- Passwords
- Personal Identifiable Information (Privacy)
- Payment Information
- Sensor Integrity/ Application Integrity

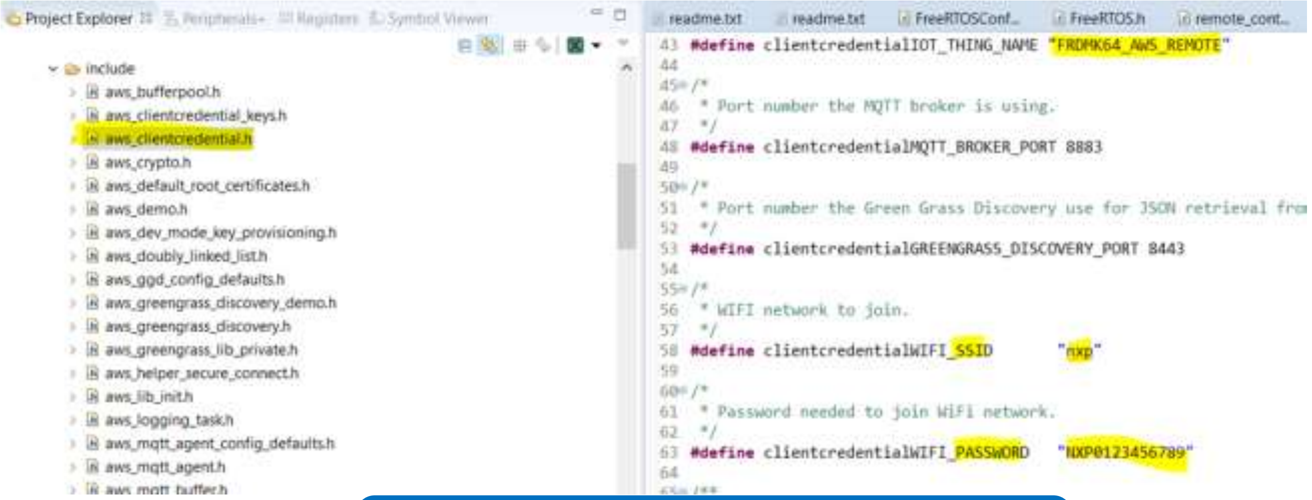
Requirements for IoT Data

- Integrity
- Authenticity
- Availability (Conditional)[1)]
- Confidentiality (Conditional)[1)]

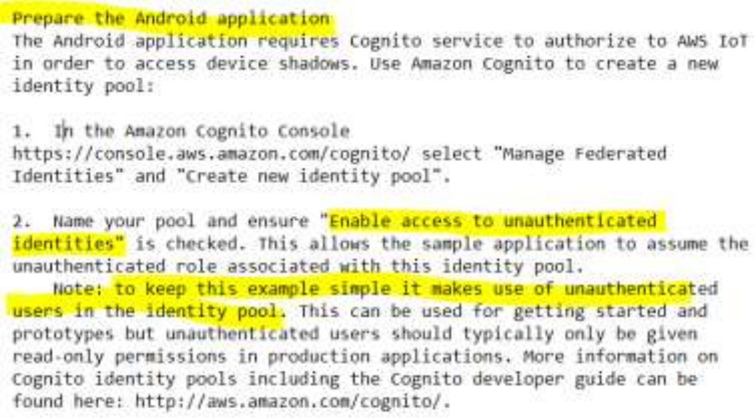
1) Conditional requirements depend on the device – they are not always required



Protect device private keys with encryption/software



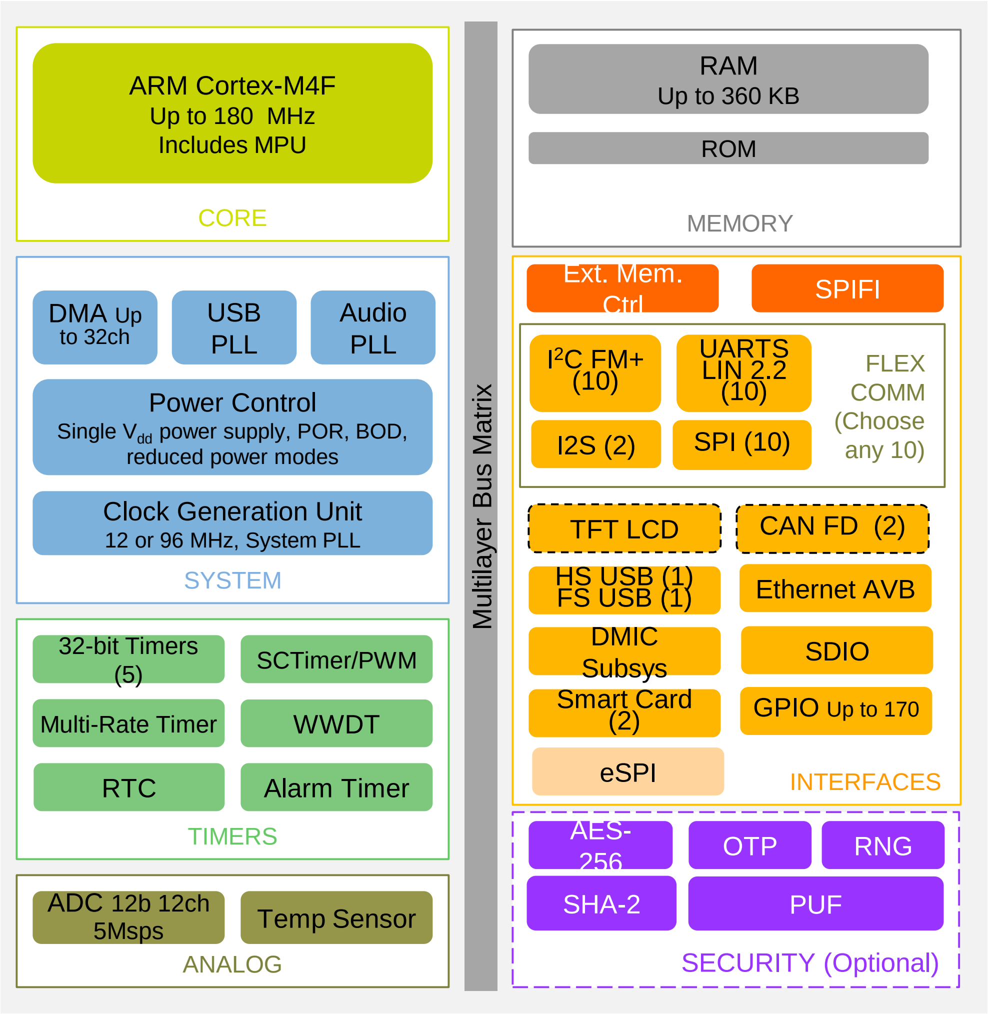
Only store encrypted WiFi credentials



Set real AWS policies

LPC54S0xx: Protecting Cloud On-boarding

LPC540xx/LPC54Sxx Block Diagram



High performance with high-end Graphical User Interface and security

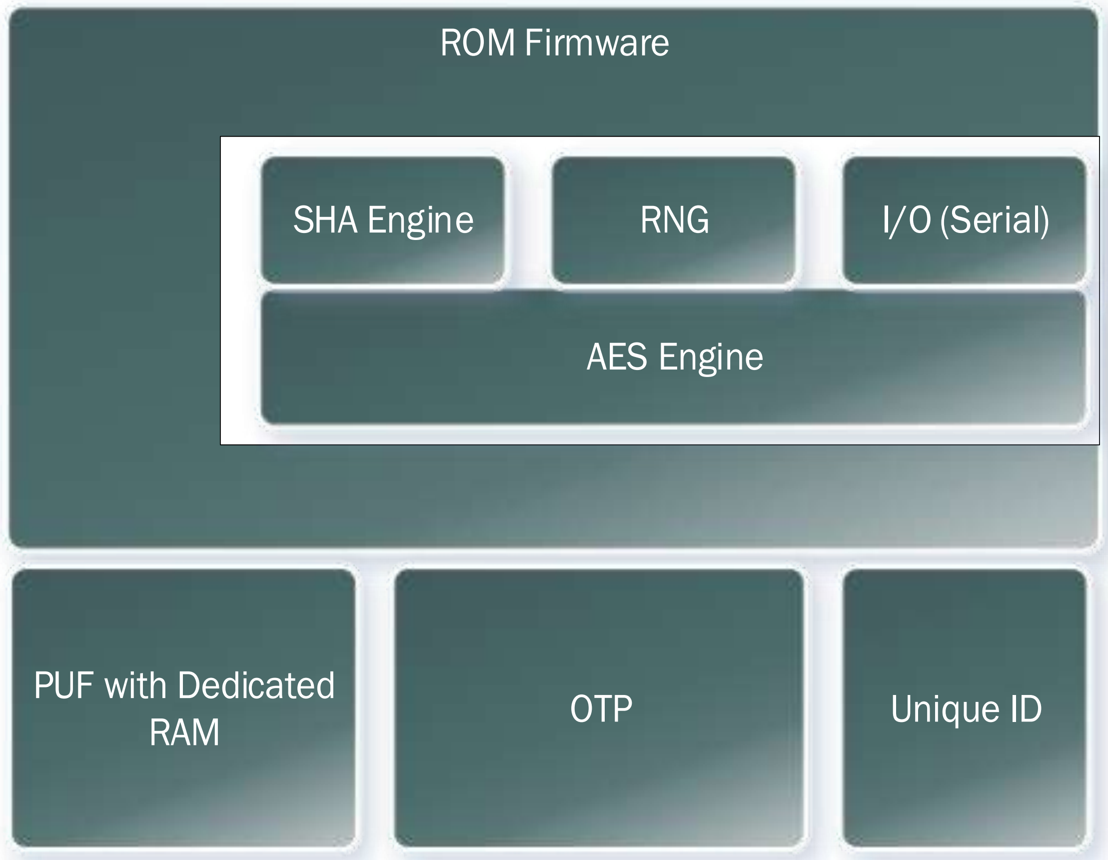
- Cortex-M4F, 180MHz
  - Up to 360 KB RAM
  - 16KB EEPROM
  - XIP from QSPI via SPIFI
  - External Memory Ctrl (up to 32 bits)

Key Features

- Graphic LCD with resolutions up to 1024 x 768
- CAN-FD controller x2 (LPC54608)
- eSPI interface (slave and LPC bus device functionality)
- Digital mic subsystem supporting voice detection
- Hi-Speed and Full Speed USB
  - USB: 1x HS (H/D) w/on-chip HS PHY
  - XTAL-less FS USB Dev
- FlexComm: flexible serial connectivity
- Advanced Security Option:
  - AES-256, SHA-2, True RNG
  - PUF for key storage
  - HW diversified OTP Key Storage
  - Secure boot using 2048-bit RSA authentication and SHA-2 verification
  - Encrypted boot using AES-GCM mode

LPC54Sxx Security Sub-system

- ROM supporting secure boot methods
  - Authentication, Encryption, combination options
- AES Engine
  - supports 128, 192 or 256 bit keys
  - Encryption modes: Electronic Codebook (**ECB**), Cipher Block Chaining (**CBC**), Cipher Feedback (**CFB**), Output Feedback (**OFB**), Counter (**CTR**), Galois Counter Mode (**GCM**)
- SHA Engine
  - Support SHA1 (160 bit) and SHA2 (256 bit)
- Physically Unclonable Function (PUF)
  - Device unique root key (256 bit strength)
  - Can store key sizes 64 bit to 4096 bit
  - Index 0 Keys routed to AES engine via direct HW bus
- 128-bit UUID per device
  - RFC4122 compliant
- Random Number Generator (RNG)
  - FIPS 140-1 compliant
- HW diversified OTP key
  - Key stored in OTP is scrambled using device unique ID



SRAM PUF Technology

1

Process Variation

Naturally occurring variations in the attributes of transistors when chips are fabricated (length, width, thickness)

2

SRAM Start-up Values

Each time an SRAM block powers on the cells come up as either a 1 or a 0

3

Silicon Fingerprint

The start-up values create a 100% random and repeatable pattern that is unique to each chip

4

SRAM PUF Key (K_{PUF})

The silicon fingerprint is turned into a secret key that builds the foundation of a security subsystem

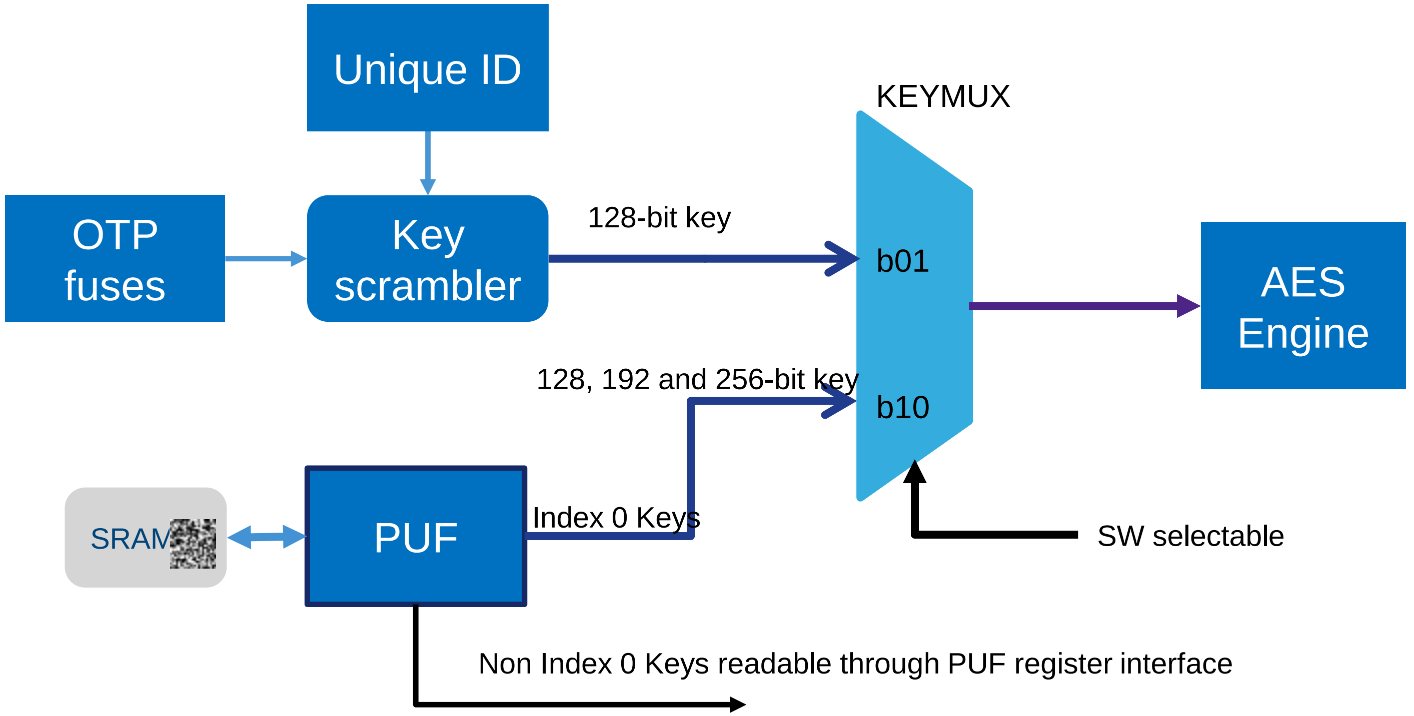
SRAM PUF Benefits

- Device-unique, non-reproducible fingerprint
- Leverages entropy of mfg. process
- No key material programmed

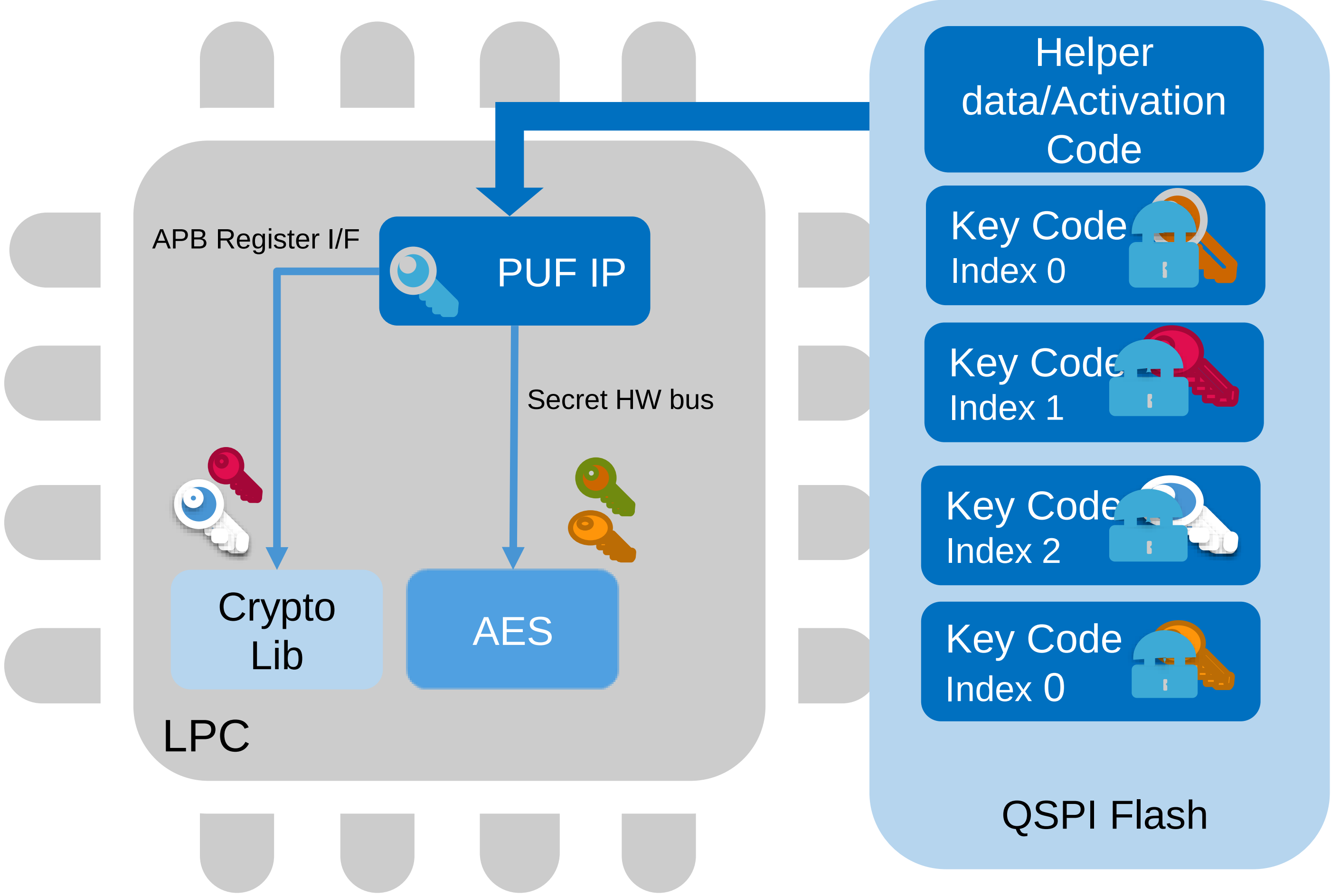
Key Management – HW AES Key Paths

- Critical keys feed directly to AES engine via HW bus
- No access to secret keys (Index 0) via SW readable registers
  - Except during provisioning

- PUF derives unique root key (K_{PUF}) per device from SRAM fingerprint
  - Eliminates complexity of generating unique keys per device during provisioning
  - Protects credentials on a per device basis



PUF Key Store

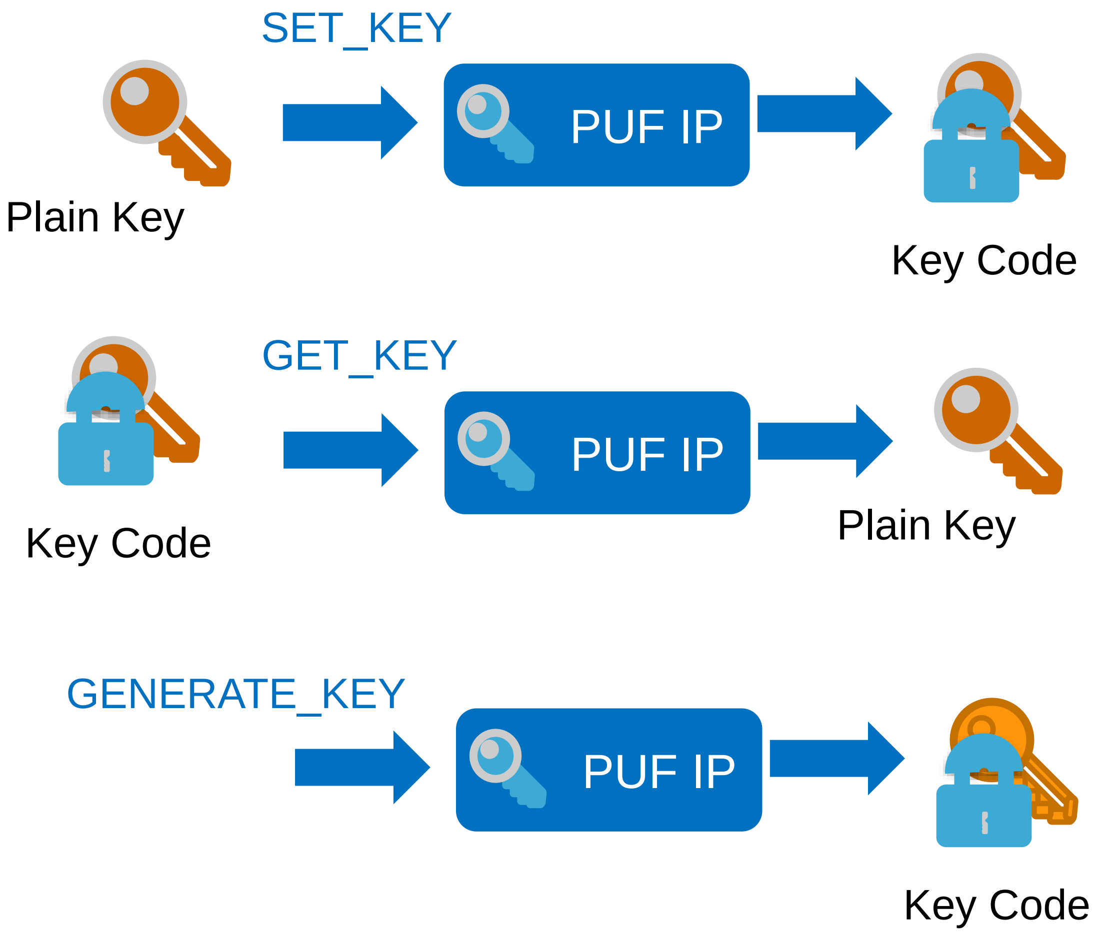


LPC PUF Features

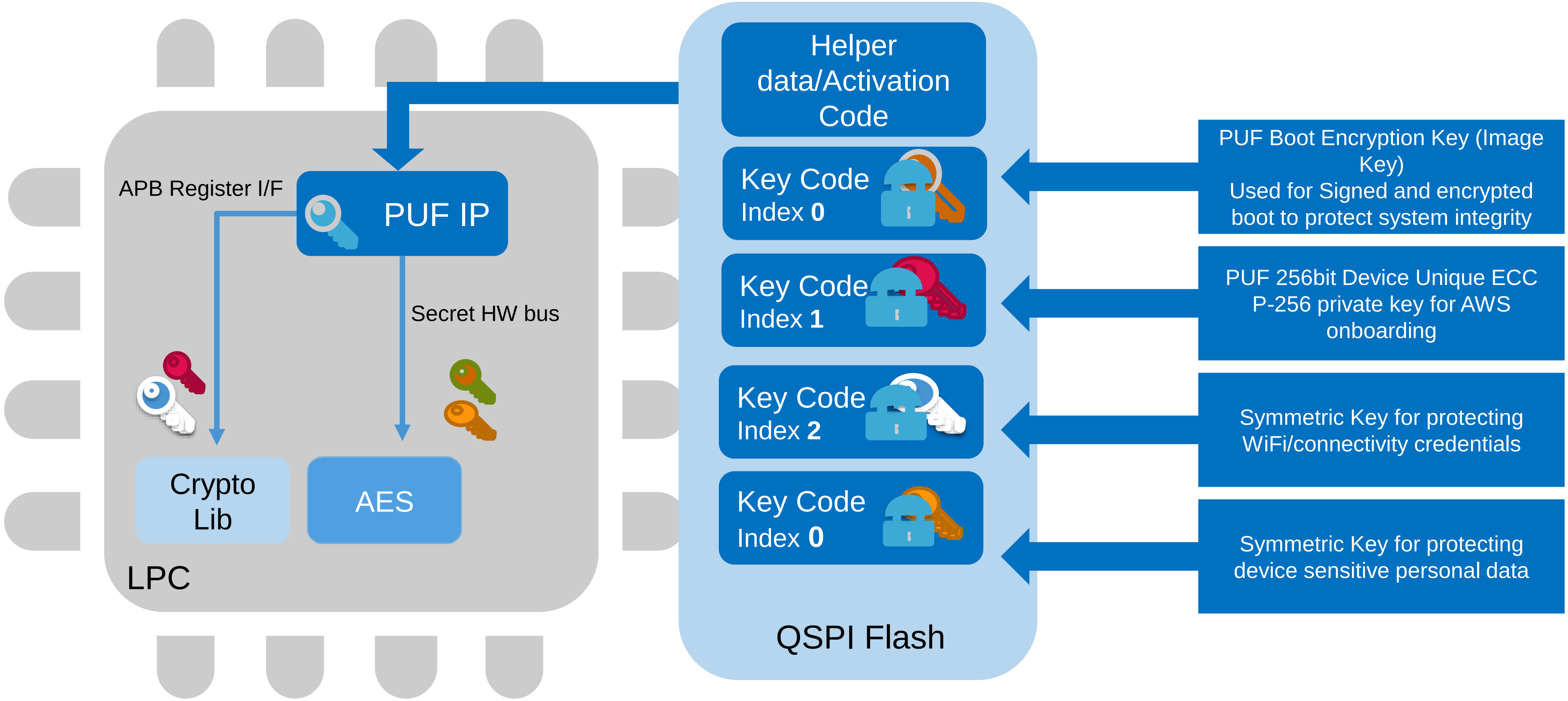
- 256 bit strength Root key
- Supports wrapping of keys
  - 64 to 4096 bits keys
  - Generation of Intrinsic keys (random key)
  - Index 0 accessible through HW secret bus
  - Other indexes through register I/F

PUF Key Store – Key Generation

- Keys generated externally can be stored through PUF using SET_KEY operation
- PUF controller provides generation of device unique cryptographic strength keys (64 to 4096 bits) using GENERATE_KEY operation
  - If key index parameter is set to 0 then key is not known to anybody.
  - Any other key index are accessible through register interface using GET_KEY operation.



PUF Key Store for Updated Amazon FreeRTOS Example



PUF Key Store for Updated Amazon FreeRTOS Example

Helper data/Activation Code

Key Code Index **0**

Key Code Index **1**

Key Code Index **2**

Key Code Index **0**

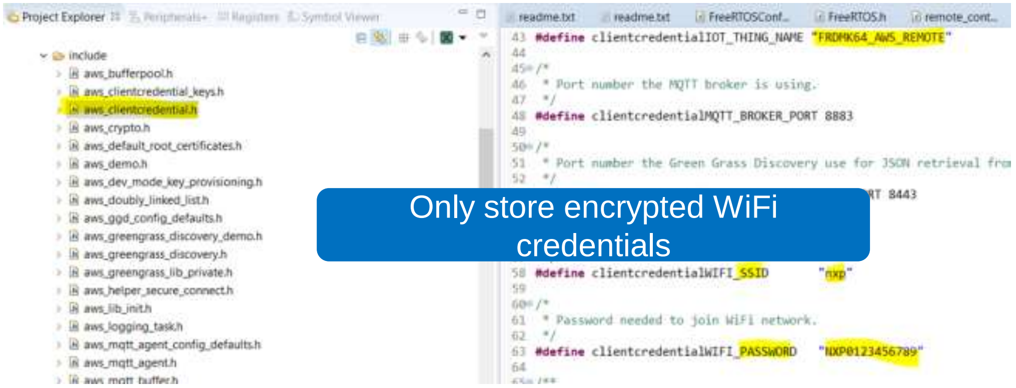
QSPI Flash

PUF Boot Encryption Key (Image Key) Used for Signed and encrypted boot to protect system integrity

PUF 256bit Device Unique ECC P-256 private key for AWS onboarding

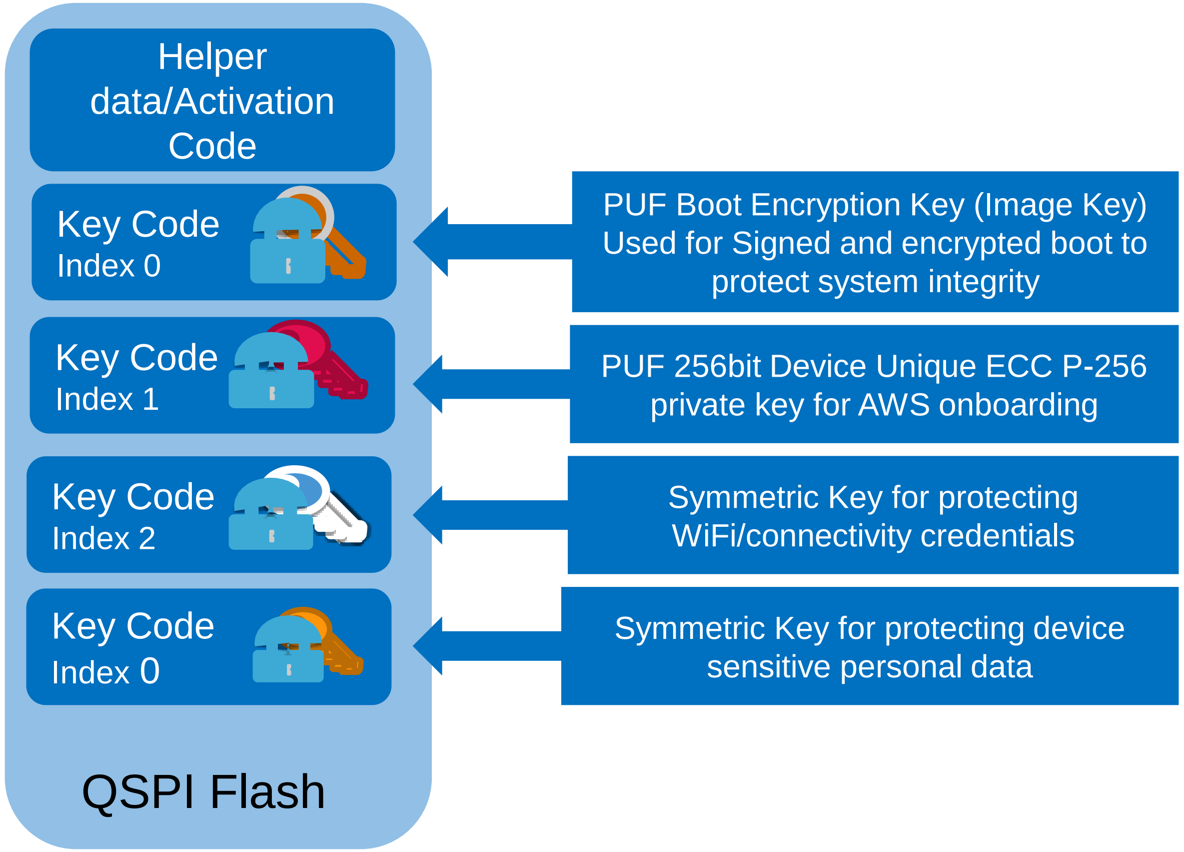
Symmetric Key for protecting WiFi/connectivity credentials

Symmetric Key for protecting device sensitive personal data



Ex: PUF Key Store for Amazon FreeRTOS and Onboarding

Key storage protected by the “PUF protected” private key



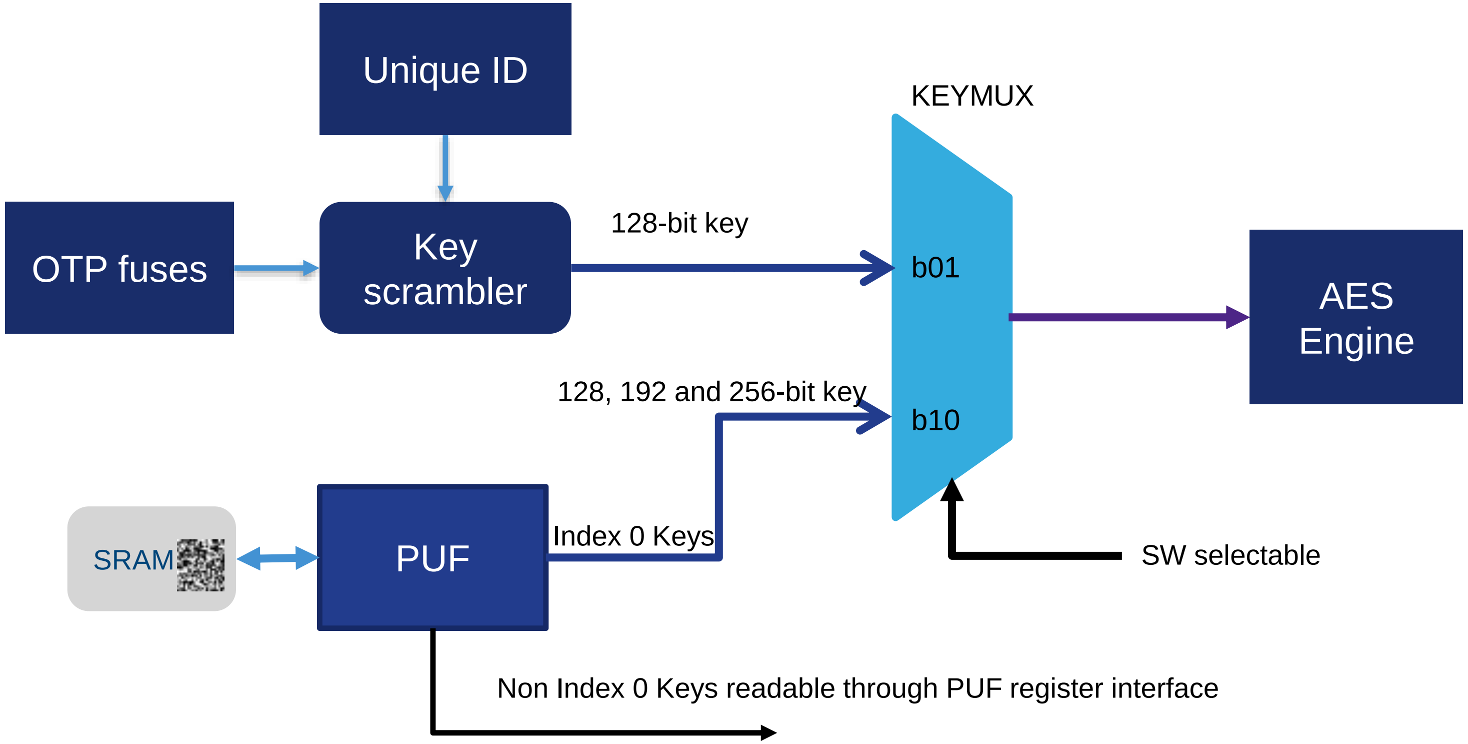
Benefits of this to onboard securely to AWS cloud

+ Protect device private key needed for AWS onboarding with a device unique symmetric key
+ Integrity is protected with the secure boot with image encryption based on PUF (physically unclonable function)
+ PUF encrypted WiFi credentials to minimize risk of WiFi access code theft

Key Management – HW AES Key Paths

- Critical keys feed directly to AES engine via HW bus
- No access to secret keys (Index 0) via SW readable registers
  - Except during provisioning of installed key
  - For PUF generated key, no access

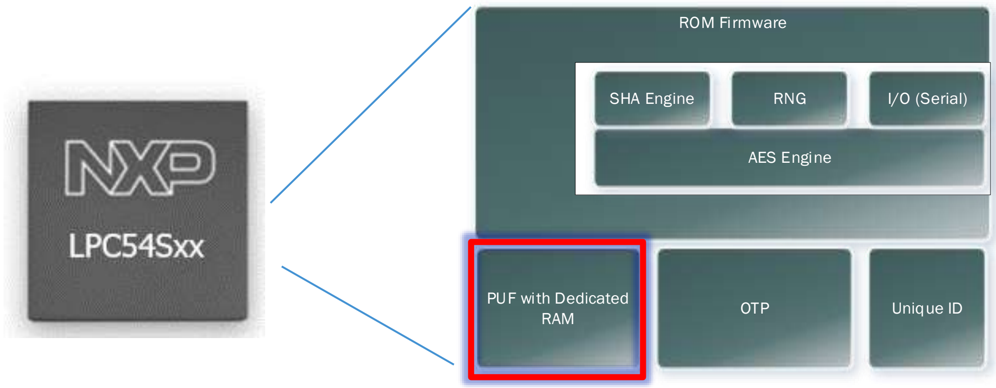
- PUF derives unique root key (K_{PUF}) per device from SRAM fingerprint
  - Eliminates complexity of generating unique keys per device during provisioning
  - Protects credentials on a per device basis



Secure Onboarding with LPC54Sxx

Key Store for Amazon FreeRTOS and onboarding

LPC54S



Security use-cases features enabled

- Enable Amazon FreeRTOS and secure onboarding to the AWS cloud, by having the key encrypted with a PUF-protected encryption
- Securely store multiple private keys to protect system data (WiFi Credentials)
- Secure boot of the device using a PUF encrypted key

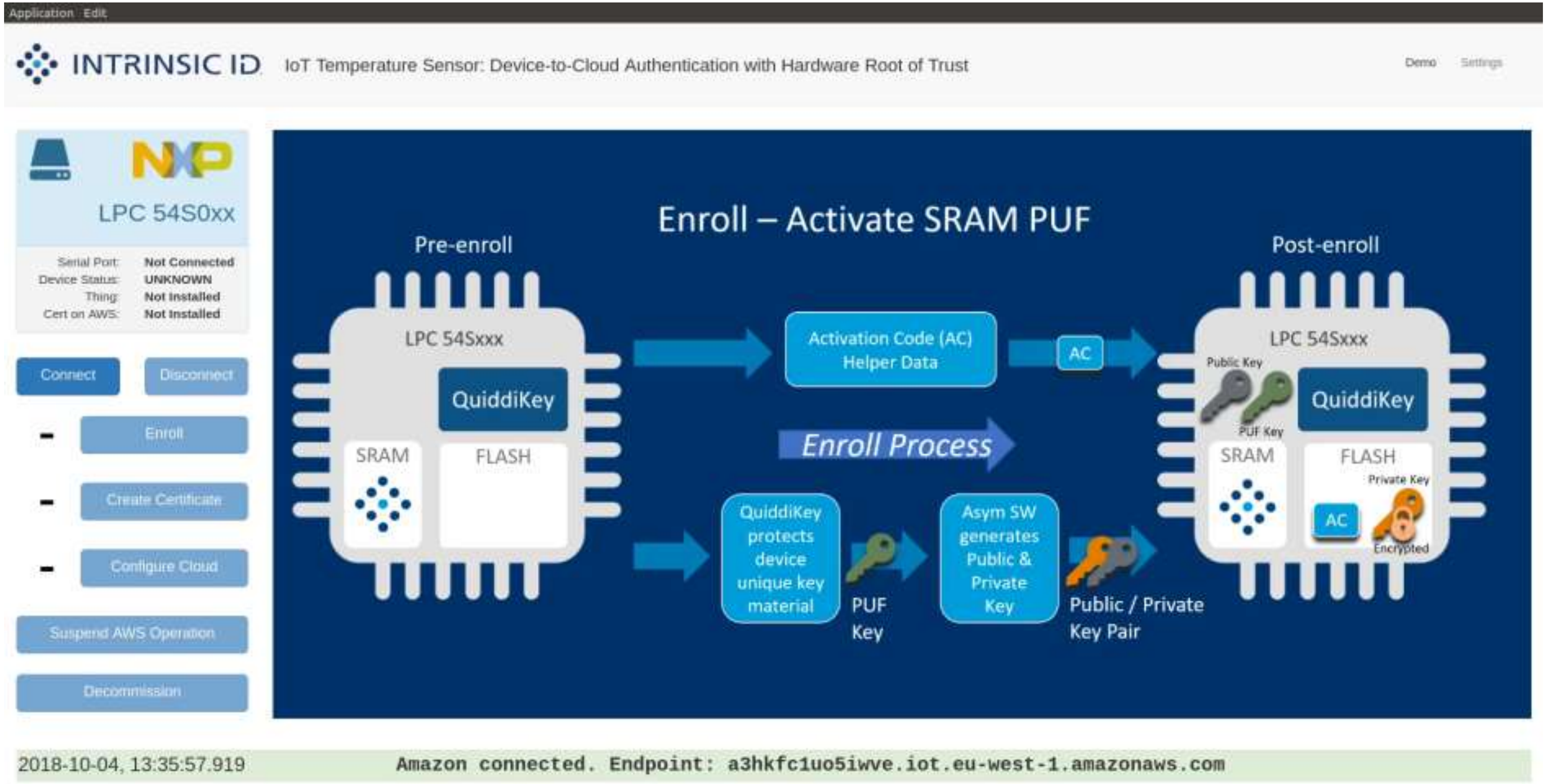
NXP products used

- LPC54Sxx IoT Microcontroller

Security features of products used

- PUF – Physically unclonable function with dedicated RAM
- HW accelerated encryption (AES, SHA) secure bus to PUF key
- ROM supporting secure boot methods

Demonstration Video



SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2019 NXP B.V.