CRYPTO QUANTIQUE

Using Cloud-based Tools to Simplify Post-Quantum IoT Security from Chip to Cloud

Q4 2024

Your speaker today

Chris Jones

Director of Applications

Chris Jones is Crypto Quantique's IoT Director of Applications. Following a 28-year career in project management and field applications engineering with Cypress Semiconductor, Renesas, and Secure Thingz, Chris joined Crypto Quantique in May 2020. Chris holds a BSc in Electrical and Electronic Engineering from the University of Coventry, UK.

CRYPTO QUANTIQUE

Agenda

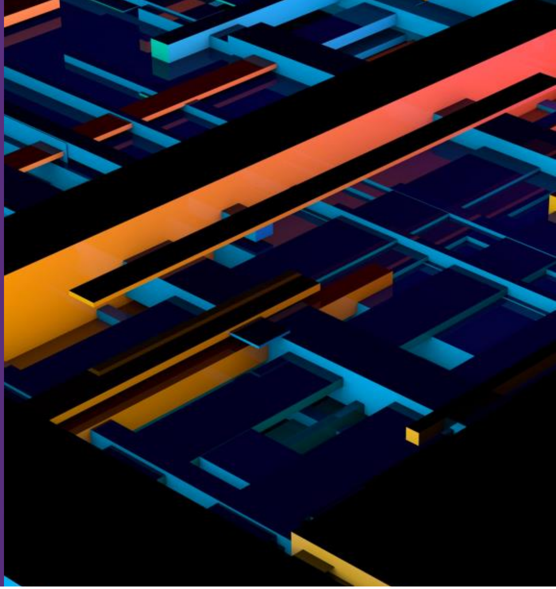
- Introduction to CQ
- Legislation
- Secure by Design
- Security Functions
- Root of Trust
- PUF Technology
- Connecting to the Cloud
- Summary
- QuarkLink Demo

CRYPTO QUANTIQUE

CRYPTO QUANTIQUE

We are a software and IP company exponentially transforming IoT cybersecurity

A team of experts in cryptography, IC design, cloud-based software development and quantum physics



Based in London, UK
Founded in 2016

The Pandemic ramped up connectivity & size of attack surface

75B

IoT devices by 2025

Cybercrime is growing exponentially

$5.5T

Cost to global economy in 2021

But IoT security can no longer be ignored

The market has reached an inflection point and manufacturers have to act

Regulation is exploding globally

20+

countries inc. US, UK, EU now working on IoT security regulations

Onboarding 1000 devices manually can take up to

2 Years

(Kaiser)

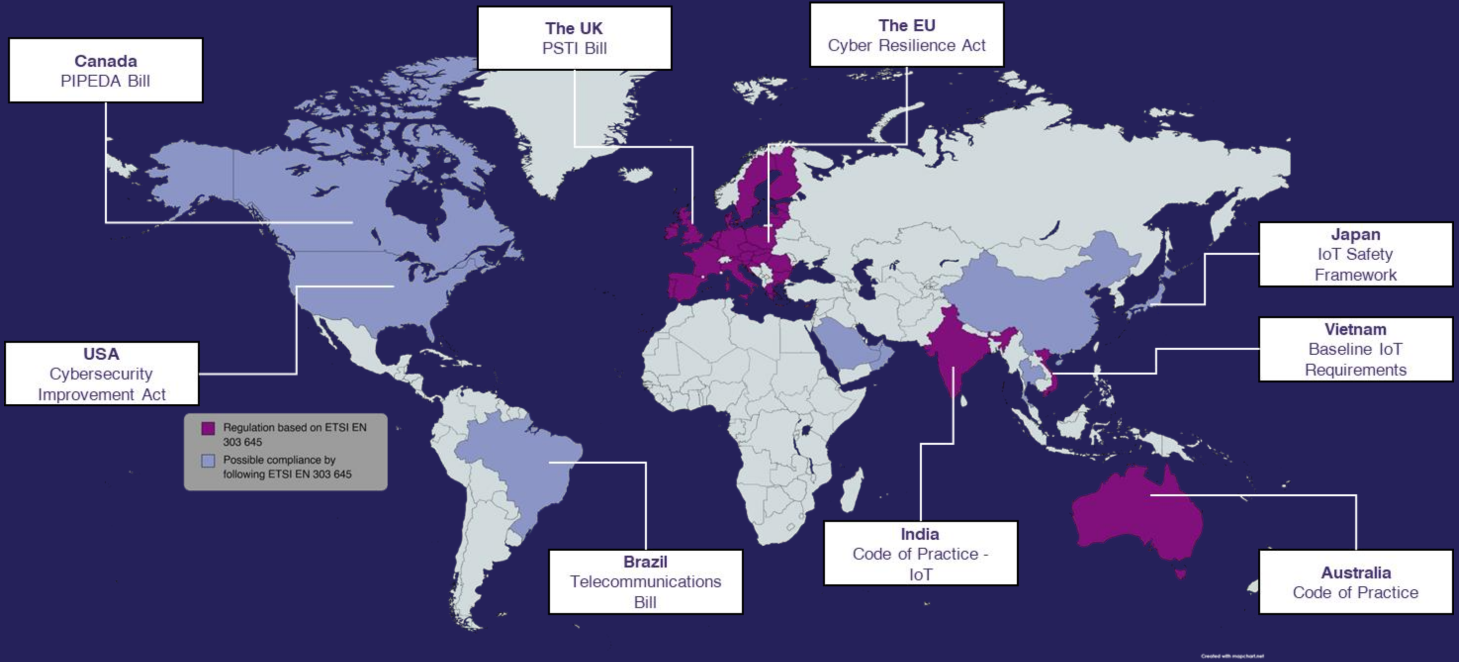
CRYPTO QUANTIQUE

Legislation

Section 1

Legislation/Regulation

Forces the security burden from consumers to IoT manufacturers and service providers.



CRYPTO QUANTIQUE

Source - CETOME

EU Cyber Resilience Act - An overview

Scope of Cyber Resilience Act

What does it cover?

Products with digital elements:

1. Hardware products and components placed on the market separately, such as laptops, smart appliances, mobile phones, network equipment or CPUs

2. Software products and components placed on the market separately, such as operating systems, word processing, games or mobile apps

What is not covered:

❖ Non-commercial projects, or services that are covered by other regulations

❖ Already regulated areas such as cars, medical devices, and aeronautical equipment

CRYPTO QUANTIQUE

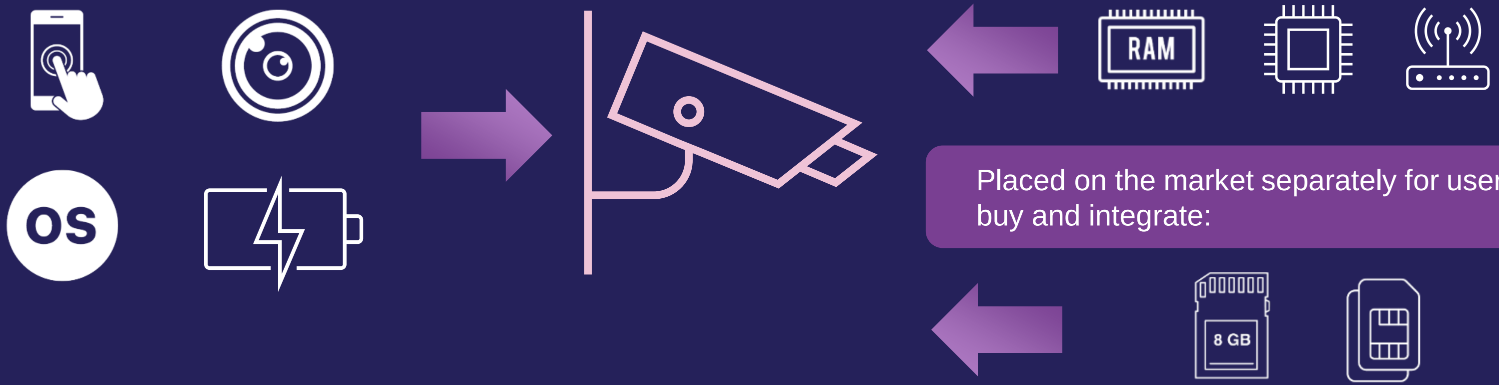
A simplified example application

As a rule, whoever places on the market a “final” product or a component is required to comply with the essential requirements, undergo conformity assessment and affix the CE marking.

Developed by the manufacturer placing the product i.e., IP Camera on the market:

Developed by upstream manufacturers for integration into the “final” product:

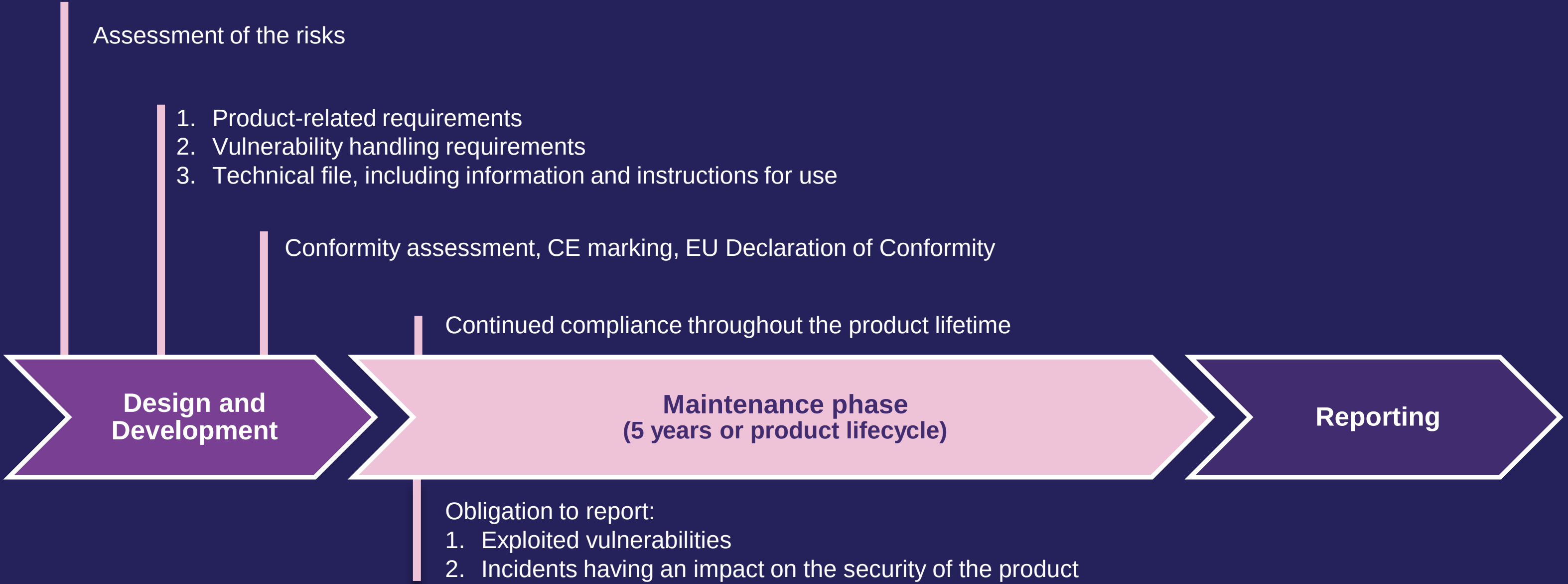
Placed on the market separately for users to buy and integrate:



CRYPTO QUANTIQUE

Obligations of manufacturers

What will manufacturers have to do under the new regulations



Assessment of the risks

1. Product-related requirements
2. Vulnerability handling requirements
3. Technical file, including information and instructions for use

Conformity assessment, CE marking, EU Declaration of Conformity

Continued compliance throughout the product lifetime

Obligation to report:
1. Exploited vulnerabilities
2. Incidents having an impact on the security of the product

CRYPTO QUANTIQUE

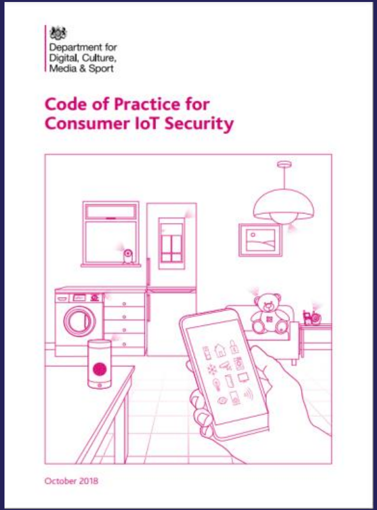
Security by Design

Section 2

Secure Design – Best Practices

CRYPTO QUANTIQUE

Government

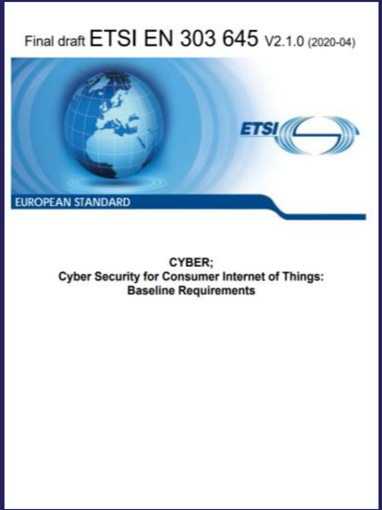

Department for Digital, Culture, Media & Sport

Code of Practice for Consumer IoT Security

October 2018

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/971440/Code_of_Practice_for_Consumer_IoT_Security_October_2018_V2.pdf

ETSI

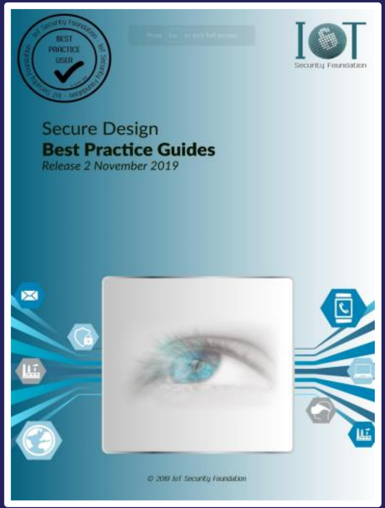

Final draft ETSI EN 303 645 V2.1.0 (2020-04)

EUROPEAN STANDARD

CYBER;
Cyber Security for Consumer Internet of Things:
Baseline Requirements

https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf

IoT Security Foundation


Secure Design
Best Practice Guides
Release 2 November 2019

© 2019 IoT Security Foundation

https://iotsecurityfoundation.org/wp-content/uploads/2019/12/Best-Practice-Guides-Release-2_Digitalv3.pdf

Key Takeaways from Best Practices

An example of the learnings in the IoTSF Secure Design Best Practices

- Classification of Data
- Physical Security
- Secure Boot
- Secure OS
- Encryption
- Network Connections
- Secure Software Updates

CRYPTO QUANTIQUE

Key Takeaways from Best Practices

An example of the learnings in the IoTSF Secure Design Best Practices

- Classification of Data
- Physical Security
- Secure Boot
- Secure OS
- Encryption
- Network Connections
- Secure Software Updates

CRYPTO QUANTIQUE

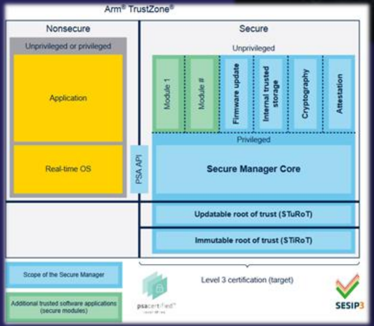
Security Functions

Section 3

Security is complex

- The implementation of security functions in a connected device is highly complex. Expertise in embedded firmware/software engineering, cryptography, provisioning and remote communications is necessary.

- Semiconductor vendors have been trying to simplify the processes and tools used develop a secure application for many years with mixed results.


Arm® TrustZone®
Nonsecure
Secure
Unprivileged or privileged
Unprivileged
Application
Module 1
Module #
Firmware update
Internal trusted storage
Cryptography
Attestation
Privileged
Real-time OS
PSA API
Secure Manager Core
Updatable root of trust (STuRoT)
Immutable root of trust (STiRoT)
Scope of the Secure Manager
Additional trusted software applications (secure modules)
Level 3 certification (target)
SESIP3

CRYPTO QUANTIQUE

Cryptography

- The science of exchanging information in plain sight under the presence of malicious adversaries.
- Adversary’s main goal is typically to either read hidden communications or disrupt messages.
- Best practice is to assume adversary understands techniques used to hide message, so security must be derived from some hidden information, referred to as a key.
- If both parties share a common key, we say that we are doing symmetric cryptography.
- If both parties are unable to agree on a common key before beginning communications, we say we are doing asymmetric cryptography.
- Techniques for asymmetric cryptography are substantially slower than their symmetric counterparts, but in practice there is need for both.

CRYPTO QUANTIQUE

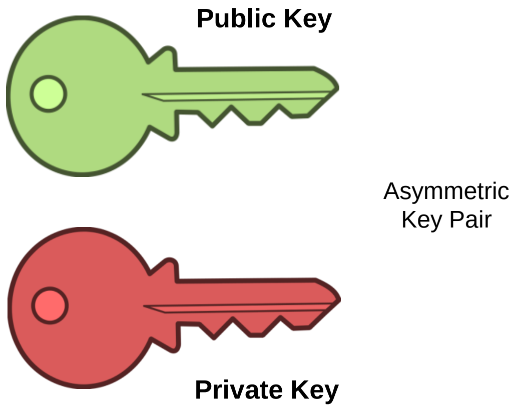
Asymmetric Ciphers

- Asymmetric ciphers are based on a “difficult-to-solve” problem forming a one-way function e.g. RSA is based on the factorization of large prime numbers

- Asymmetric ciphers always use a **key pair** consisting of a **Private Key** and a corresponding **Public Key**. Properties of this key pair are :

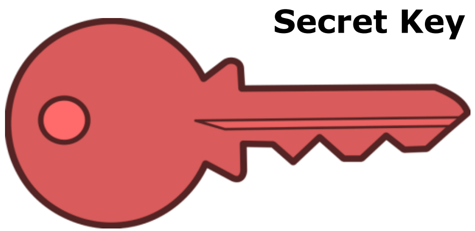
  - Either key can encrypt information which can be decrypted by the other key
  - Either key can be used to sign some data which can be verified by the other key

- However, the **Private Key** **must never be disclosed** and is in many cases assigned to an identity, person or thing



Symmetric Ciphers

- Symmetric ciphers are suitable for encrypting large portions of data
- Sender and receiver share one **Secret Key**, which must be protected on both ends
- Symmetric ciphers can be either operated as streams or block ciphers on basis of well-known algorithms e.g. AES-CBC or AES-ECB
- Symmetric ciphers were invented before asymmetric ciphers



Cryptographic Key comparison

Security Level (bits)	Symmetric Key Size (bits)	ECC Key Size (bits)	RSA Key Size (bits)	Post-Quantum Key Size (bits)
80	80	160	1024	**Lattice-based (Kyber):** 1536
112	112	224	2048	**Lattice-based (Kyber):** 3072
128	128	256	3072	**Code-based (McEliece):** ~524288
192	192	384	7680	**Multivariate-based:** ~6000-10000
256	256	512	15360	**Hash-based (SPHINCS+):** 64 KB

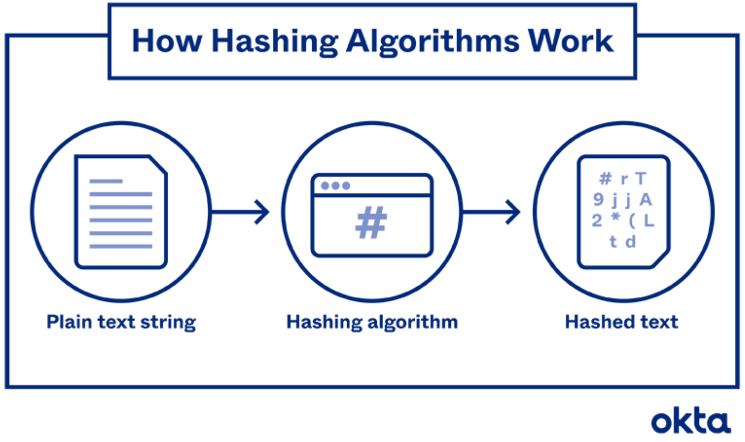
In the context of cryptography, **security level (in bits)** represents the computational difficulty, or work factor, required to break a cryptographic algorithm. It essentially defines how many operations (usually expressed in binary bits) an attacker would need to perform to compromise the system.

Example: **128-bit security level** means that it would require roughly 2^{128} operations (brute-force attempts) to break the cryptographic system. This is considered infeasible with current computing power, providing strong security.

In the context of **quantum computing** schemes could be more vulnerable for example RSA-2048 could be reduced to a security level of 56 bits (instead of 112) using Shor’s algorithm.

HASH Function

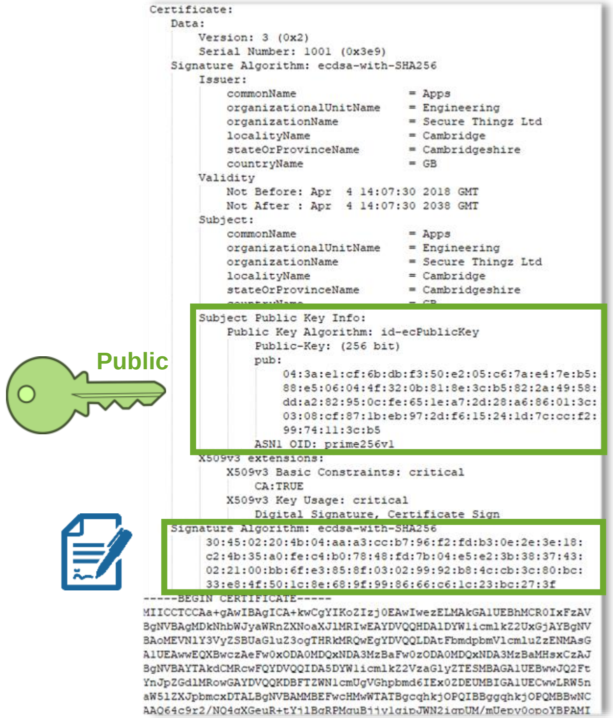
- Hashing are required for digital signatures and authentication
- Hashing is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash) and is designed to be a one-way function, that is, a function which is infeasible to invert
- FIPS 180[1] specifies the SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 hash functions



1 – Federal Information Processing Standards 180
Available from - http://dx.doi.org/10.6028/NIST.FIPS.180-4

Digital Certificates

- An example of an X.509 Certificate is shown opposite
- The certificate holds the **Public Key** of the entity, be it a Secure Device (MCU or Element), Certificate Authority, OEM or Semiconductor manufacturer
- The certificate **authenticates** the Public Key of the entity
- The Certificate is signed using the Certificate Authorities' Private Key
- The Certificate is issued by a Certificate Authority which forms part of the Chain of Trust

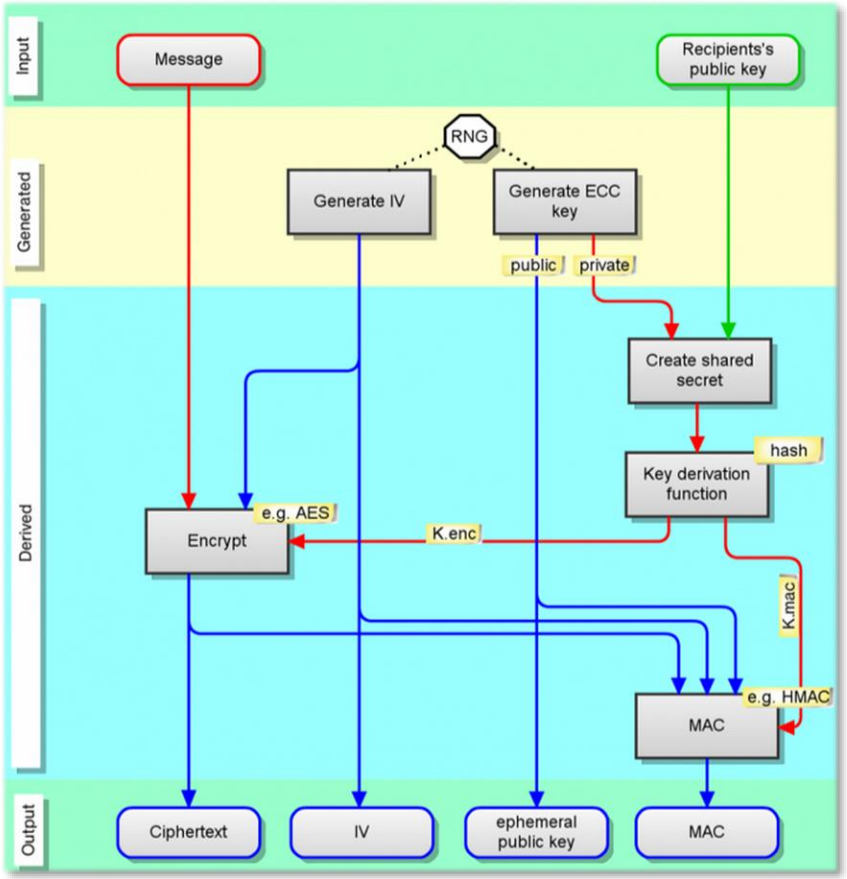


```
Certificate:
   Data:
       Version: 3 (0x2)
       Serial Number: 1001 (0x3e9)
   Signature Algorithm: ecdsa-with-SHA256
       Issuer:
           commonName                = Apps
           organizationalUnitName    = Engineering
           organizationName          = Secure Thingz Ltd
           localityName              = Cambridge
           stateOrProvinceName       = Cambridgeshire
           countryName               = GB
       Validity
           Not Before: Apr  4 14:07:30 2018 GMT
           Not After : Apr  4 14:07:30 2038 GMT
       Subject:
           commonName                = Apps
           organizationalUnitName    = Engineering
           organizationName          = Secure Thingz Ltd
           localityName              = Cambridge
           stateOrProvinceName       = Cambridgeshire
           countryName               = GB
       Subject Public Key Info:
           Public Key Algorithm: id-ecPublicKey
               Public-Key: (256 bit)
               pub:
                   04:3a:e1:cf:6b:db:f3:50:e2:05:c6:7a:e4:7e:b5:
                   88:e5:06:04:4f:32:0b:81:8e:3c:b5:82:2a:49:58:
                   dd:a2:82:95:0c:fe:65:1e:a7:2d:28:a6:86:01:3c:
                   03:08:cf:87:1b:eb:97:2d:f6:15:24:1d:7c:cc:f2:
                   99:74:11:3c:b5
               ASN1 OID: prime256v1
       X509v3 extensions:
           X509v3 Basic Constraints: critical
               CA:TRUE
           X509v3 Key Usage: critical
               Digital Signature, Certificate Sign
   Signature Algorithm: ecdsa-with-SHA256
        30:45:02:20:4b:04:aa:a3:cc:b7:96:f2:fd:b3:0e:2e:3e:18:
        c2:4b:35:a0:fe:c4:b0:78:48:fd:7b:04:e5:e2:3b:38:37:43:
        02:21:00:bb:6f:e3:85:8f:03:02:99:92:b8:4c:cb:3c:80:bc:
        33:e8:4f:50:1c:8e:68:9f:99:86:66:c6:1c:23:bc:27:3f
-----BEGIN CERTIFICATE-----
MIICCTCCAa+gAwIBAgICA+kwCgYIKoZIzj0EAwIwezELMAkGA1UEBhMCR0IxFzAV
BgNVBAgMDkNhbWJyaWRnZXNoaXJlMRIwEAYDVQQHDA1DYW1icmlkZ2UxGjAYBgNV
BAoMEVNlY3VyZSBUaGluZ3ogTHRkMRQwEgYDVQQLDAtFbmdpbmVlcmluZzENMAsG
A1UEAwwEQXBwczAeFw0xODA0MDQxNDA3MzBaFw0zODA0MDQxNDA3MzBaMHsxCzAJ
BgNVBAYTAkdCMRcwFQYDVQQIDA5DYW1icmlkZ2VzaGlyZTESMBAGA1UEBwwJQ2Ft
YnJpZGdlMRowGAYDVQQKDBFTZWN1cmUgVGhpbmd6IEx0ZDEUMBIGA1UECwwLRW5n
aW5lZXJpbmcxDTALBgNVBAMMBEFwcHMwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
AAQ64c9r2/NQ4gXGeuR+tYjlBgRPMguBjjvlgipJWN2igpUM/mUepv0opoYBPAMI
```

Over-the-air-updates

As an example of the complexity needed, here is a flow diagram of the steps needed to transmit an encrypted file to a recipient (basis of OTA).

The process here is called **ECIES**
(Elliptic Curve Integrated Encryption Scheme).



Root of Trust

Section 4

Secure Boot

- **Secure boot** is a process that ensures only trusted, authorized software is executed during the system startup. It checks the integrity and authenticity of the bootloader, operating system, and other critical software components before allowing them to load.

- **Secure boot** builds upon the **root of trust** by using it as the foundation to ensure that only trusted and verified code is executed at boot time. Without a root of trust, the integrity of secure boot cannot be guaranteed.

- A **Root-of-trust** is a set of ***unconditionally*** trusted functions and must be a computing engine, because it must perform actions.

- The RoT typically comprises:

  A micro-controller/processor that includes:

  - Capability to secure an area of memory (e.g.TrustZone, Flash Access Window)
    - Secure storage of cryptographic keys
  - A ROM based level one bootloader (typically programmed at the silicon level)
    - Capability to authenticate a software image prior to execution
  - Capability to disable unauthorised access via debug/JTAG ports

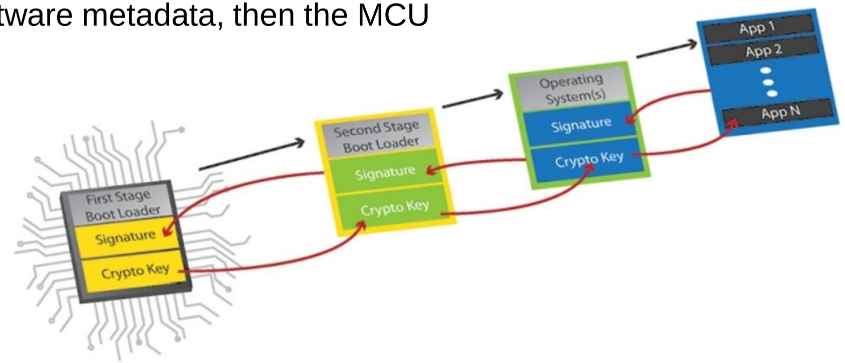
- The RoT must be **securely** provisioned (programmed) into the product

Chain of Trust

- Typically, the MCU is manufactured with a Root of Trust that has;
    1. a hard-coded public key or HASH of the public key, or **(silicon manufacturer owns the bootloader key)**
    2. a programmable OTP area that can be programmed by an OEM **(OEM owns the bootloader key)**

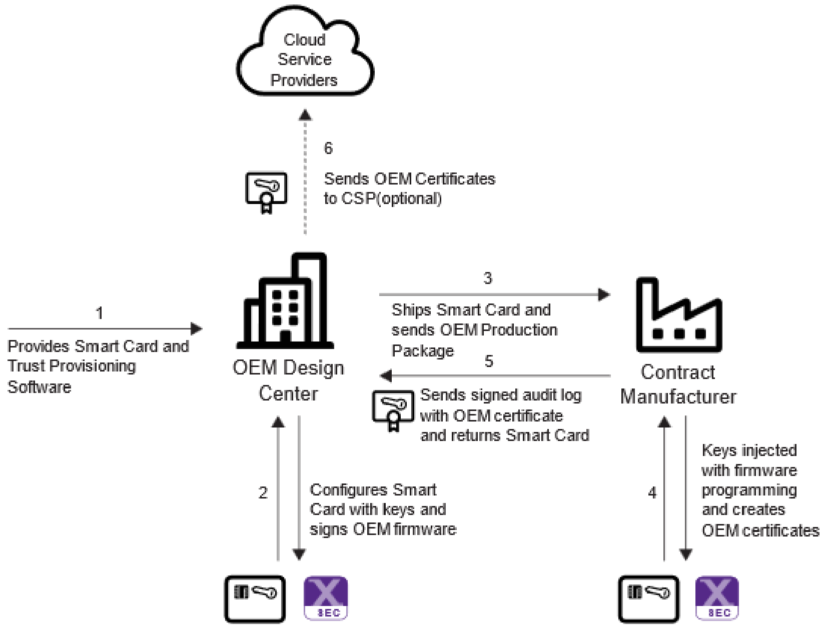
  The ROM code is immutable and will always run after reset.
  The ROM code fetches the HASH of the public key from the Root of Trust and uses it to verify the software

- The private key is used to sign the software (e.g.; next stage bootloader or OS)
- The software metadata includes a HASH of the public key (of the corresponding private key that was used to sign the software).
- If the HASH in the MCU Root of Trust matches that in the software metadata, then the MCU knows it has the correct key.
- The MCU then HASHes the software and checks the HASH in the signature using the public key
- If the HASHes match, then the software can be run.



Root of Trust Provisioning

- During manufacturing, cryptographic keys are required to be programmed into the hardware.
- Typically, these are the identity and secure boot verification keys.
- Keys are injected into the hardware during manufacture using a secure programming facility.
- Keys are generated by a **Hardware Security Module** (or Smart Card) and flashed into the hardware.
- This task is complex, expensive and not secure.


Cloud Service Providers
6
Sends OEM Certificates to CSP(optional)
1
Provides Smart Card and Trust Provisioning Software
OEM Design Center
3
Ships Smart Card and sends OEM Production Package
5
Sends signed audit log with OEM certificate and returns Smart Card
Contract Manufacturer
2
Configures Smart Card with keys and signs OEM firmware
4
Keys injected with firmware programming and creates OEM certificates
SEC
SEC

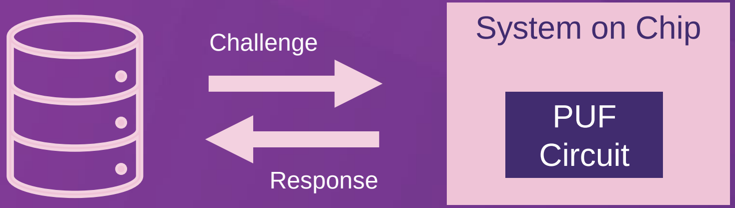
PUF Technology

Section 5

Strong vs Weak PUF

Extensive PUF (Strong PUF)

Unlimited challenge-response pairs

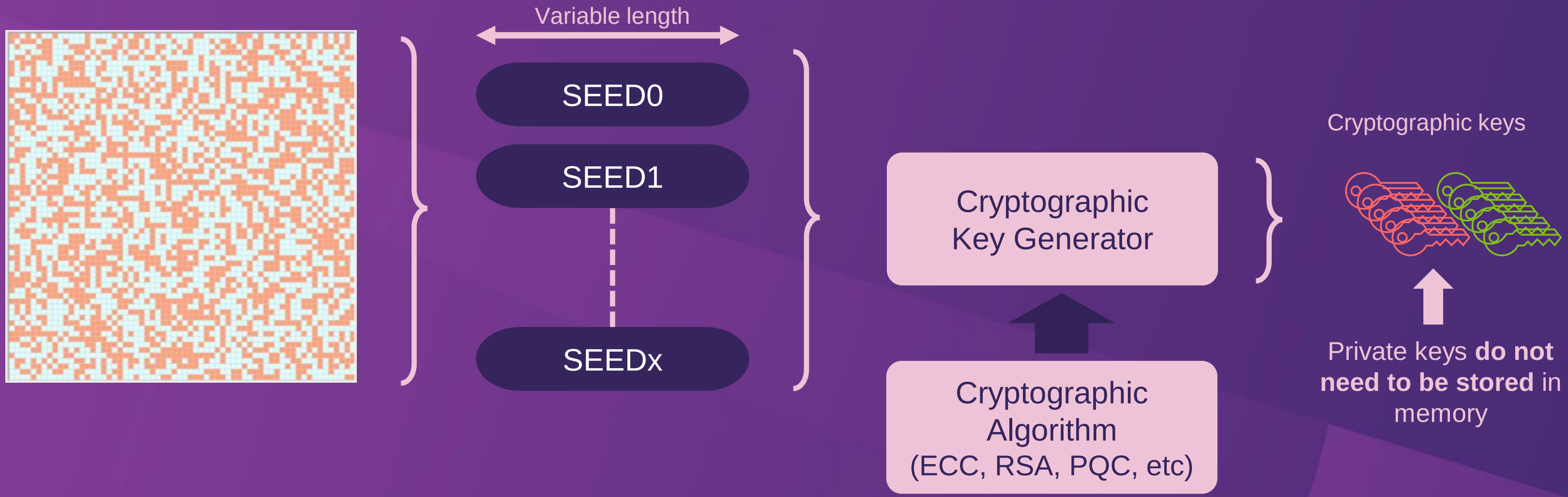


Confined PUF (Weak PUF)

A few random, unforgeable bit strings

Cryptographically agile

The PUF provides the seeds for key generation


Variable length
SEED0
SEED1
SEEDx
Cryptographic Key Generator
Cryptographic Algorithm (ECC, RSA, PQC, etc)
Cryptographic keys

Private keys **do not need to be stored** in memory

CRYPTO QUANTIQUE

Quantum Tunnelling

- Quantum tunnelling is extremely sensitive to the nanostructure of the atomic layers that make up the SiO_2 oxide
    - Make for a very good source from which to extract randomness
- Even though manufacturing processes are very tightly controlled (see figure 1), it is still impossible to control the thickness of the oxide down to the atomic level
- CQ Quantum Array consist of transistor pairs. The random difference between the insulation layers for each transistor causes two different currents which we measure with our AFE.

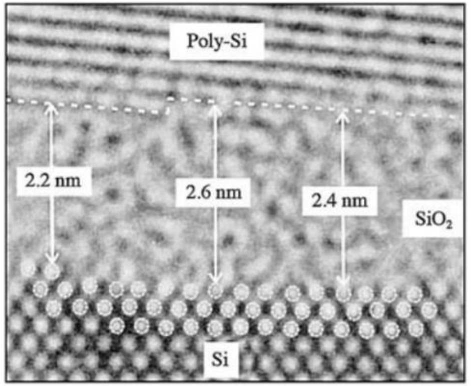
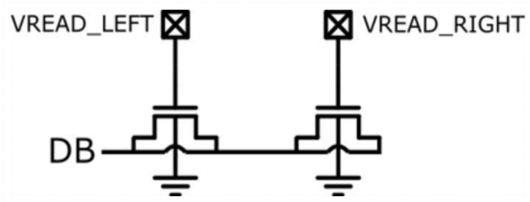


Figure 1 - SiO_2 interface roughness cross-section (IBM) [3]



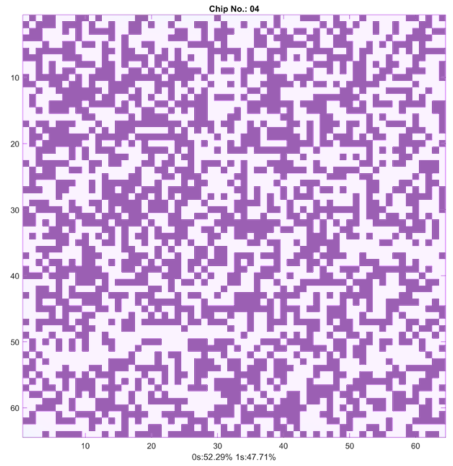
1 bit (NOT qubit)

Quantum Tunnelling (cont’d)

- Due to the inherently random nature of the atomic positions and imperfections of these nanostructures (see figure 2) it would take vast amounts of computing power to simulate

- Even with a modest quantum computer, it would be impractical to attempt to copy or simulate the device at the atomic level

Representation or fingerprint output



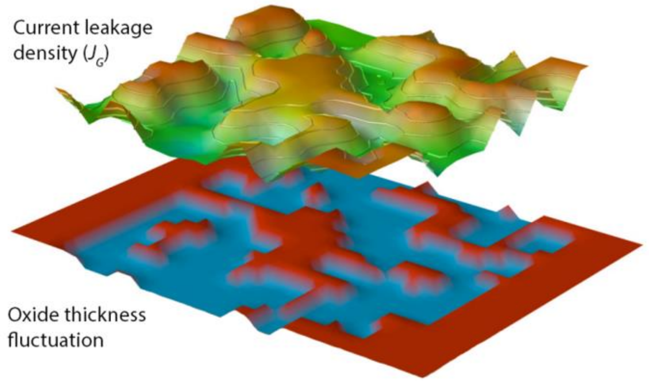
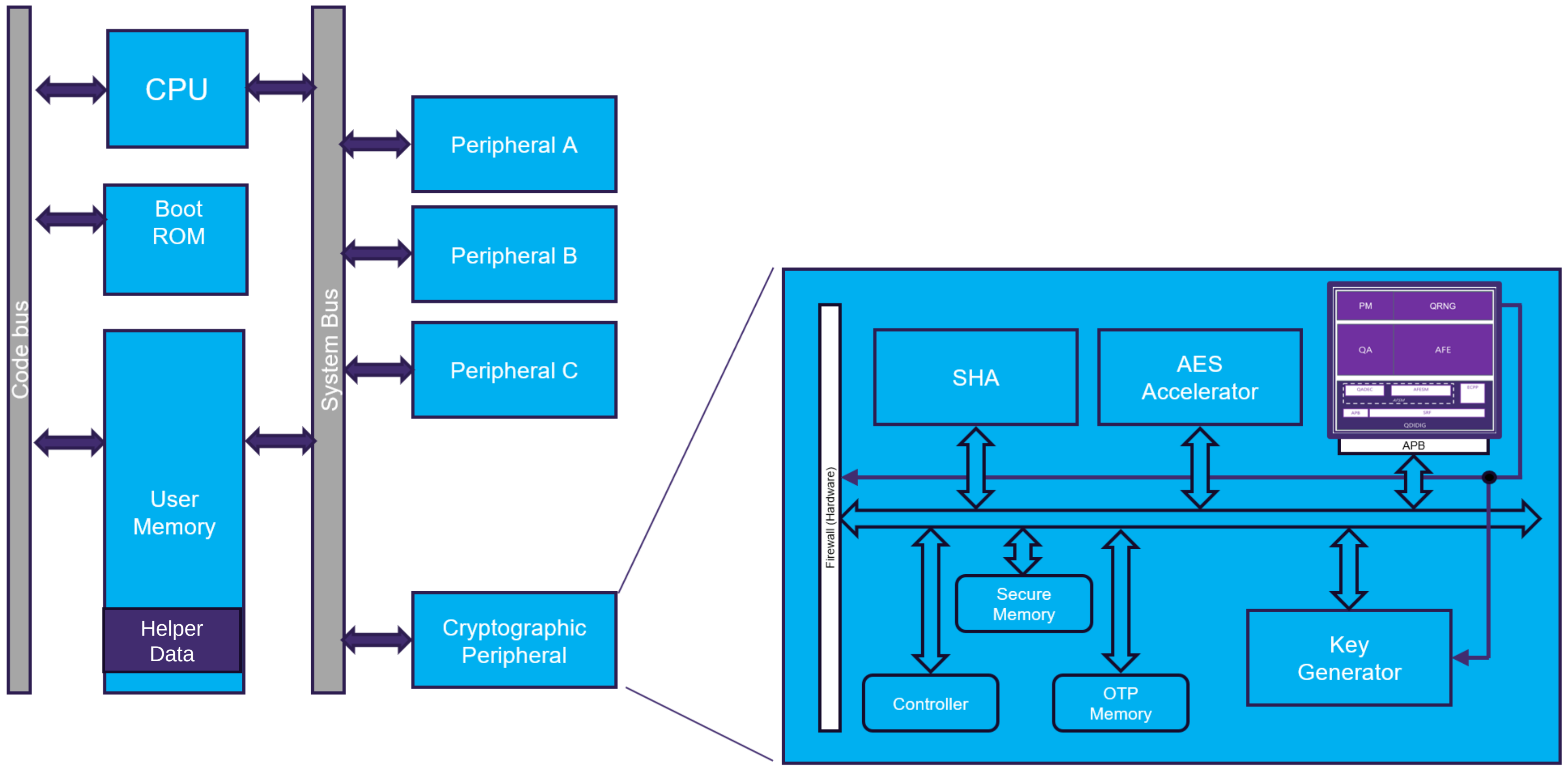


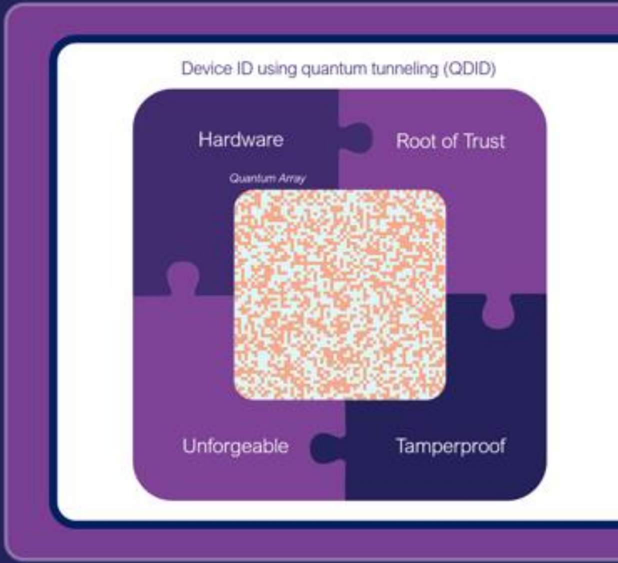
Figure 2 - Direct tunnelling current density, Si/SiO_2 interface roughness features (blue identifies regions of SiO_2 protrusions into the substrate, i.e. thicker oxide while red corresponds to thinner oxide) [4]

Typical PUF Enabled SoC (with QRNG)



Quantum Driven ID PUF

- Seeds derived from quantum tunneling sources in silicon
- Secrets derived form **QDID** are NOT stored in conventional memory, just generated on read access.
- Eliminates the cost and risk of key injection
- Low error rate, multiple uncorrelated keys
- Uses standard CMOS transistors
- High entropy, cryptographically proven randomness
- Immune to traditional side channel attacks
- No remanence effect



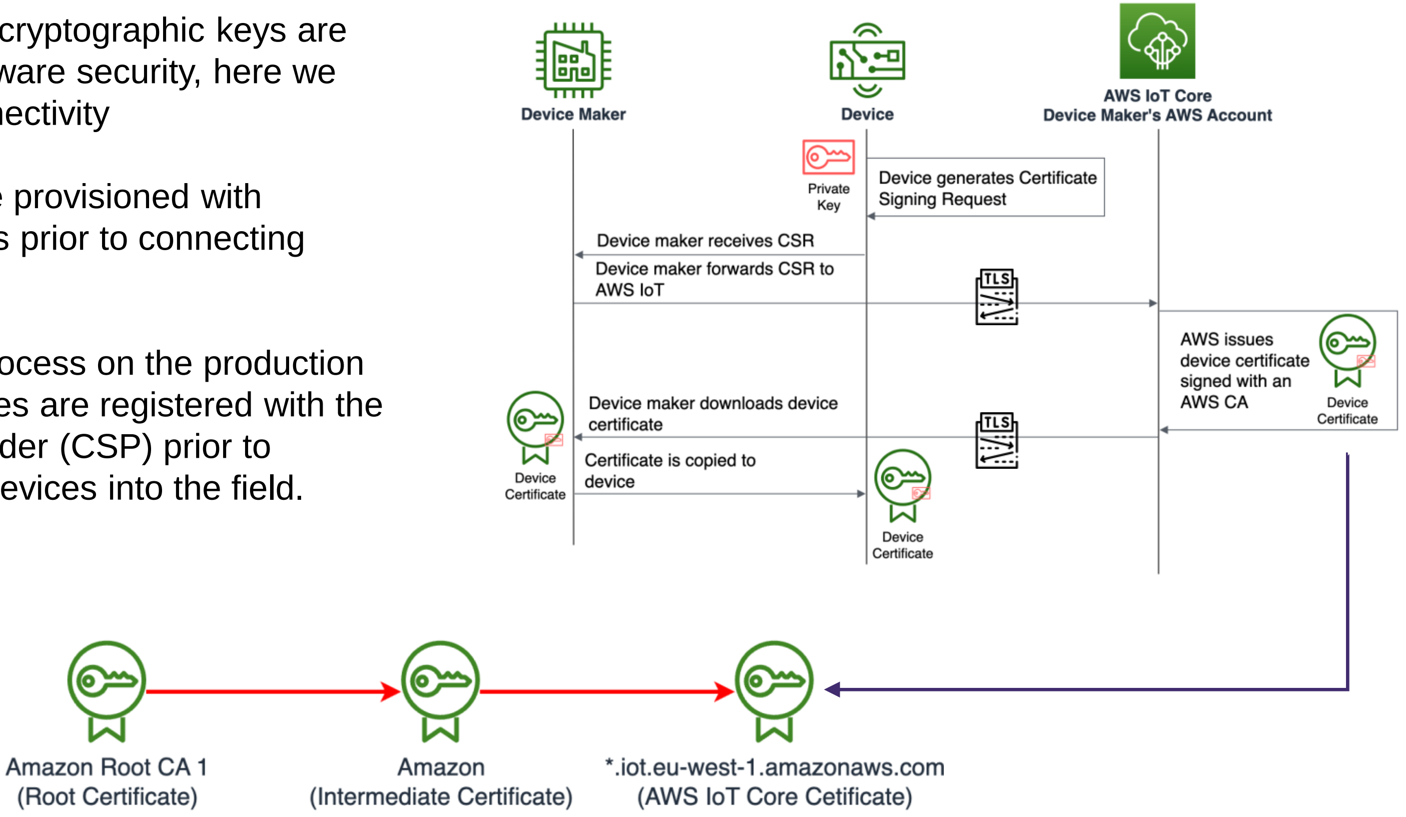
CRYPTO QUANTIQUE

Connecting to the Cloud

Section 6

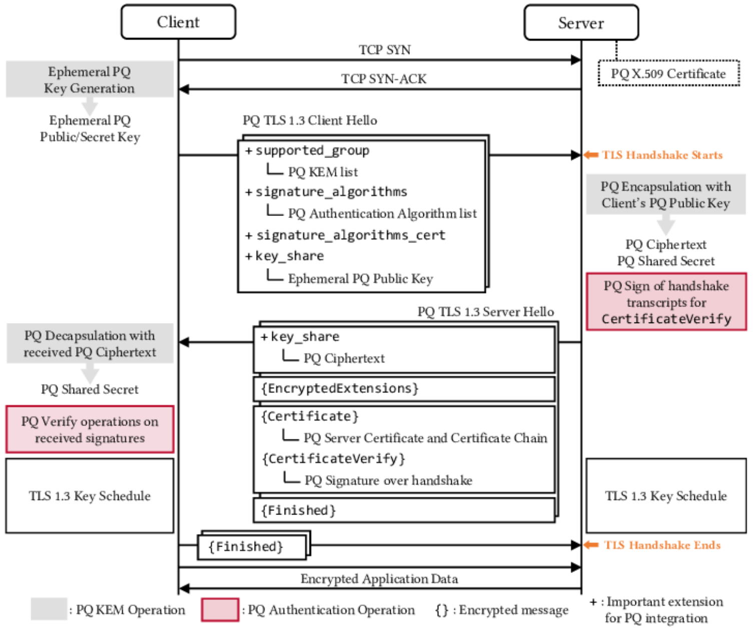
Cloud connectivity

- We have seen how cryptographic keys are used to ensure software security, here we examine cloud connectivity
- IoT devices must be provisioned with keys and certificates prior to connecting to a cloud service
- This is typically a process on the production line where certificates are registered with the Cloud Service Provider (CSP) prior to deployment of the devices into the field.



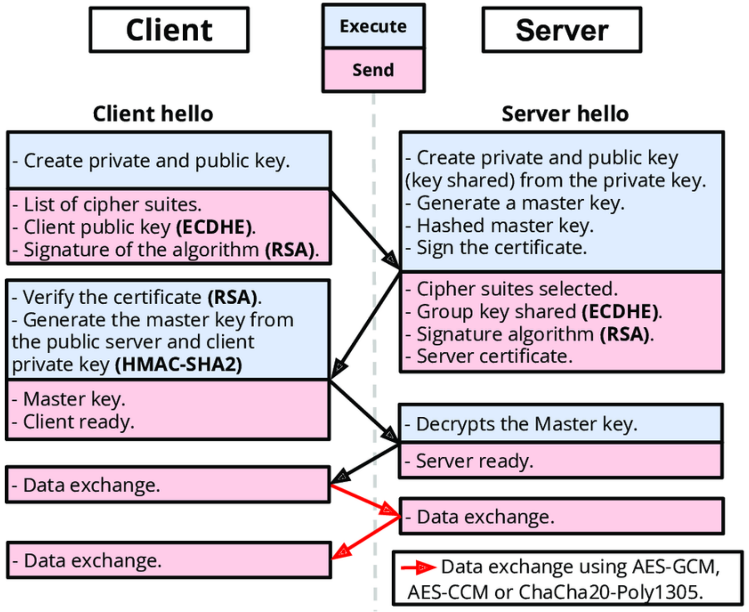
Device Maker
Device
AWS IoT Core
Device Maker's AWS Account
Private Key
Device generates Certificate Signing Request
Device maker receives CSR
Device maker forwards CSR to AWS IoT
TLS
AWS issues device certificate signed with an AWS CA
Device Certificate
Device maker downloads device certificate
TLS
Certificate is copied to device
Device Certificate
Device Certificate
Amazon Root CA 1
(Root Certificate)
Amazon
(Intermediate Certificate)
*.iot.eu-west-1.amazonaws.com
(AWS IoT Core Cetificate)

Secure connectivity (Transport Layer Security)



Client
Server
TCP SYN
TCP SYN-ACK
PQ X.509 Certificate
Ephemeral PQ Key Generation
Ephemeral PQ Public/Secret Key
PQ TLS 1.3 Client Hello
+ supported_group
PQ KEM list
+ signature_algorithms
PQ Authentication Algorithm list
+ signature_algorithms_cert
+ key_share
Ephemeral PQ Public Key
TLS Handshake Starts
PQ Encapsulation with Client's PQ Public Key
PQ Ciphertext
PQ Shared Secret
PQ Sign of handshake transcripts for CertificateVerify
PQ TLS 1.3 Server Hello
+ key_share
PQ Ciphertext
{EncryptedExtensions}
{Certificate}
PQ Server Certificate and Certificate Chain
{CertificateVerify}
PQ Signature over handshake
{Finished}
PQ Decapsulation with received PQ Ciphertext
PQ Shared Secret
PQ Verify operations on received signatures
TLS 1.3 Key Schedule
TLS 1.3 Key Schedule
{Finished}
TLS Handshake Ends
Encrypted Application Data
: PQ KEM Operation
: PQ Authentication Operation
{} : Encrypted message
+ : Important extension for PQ integration

KEM-TLS Protocol



Client
Execute
Send
Server
Client hello
Server hello
- Create private and public key.
- List of cipher suites.
- Client public key (ECDHE).
- Signature of the algorithm (RSA).
- Create private and public key (key shared) from the private key.
- Generate a master key.
- Hashed master key.
- Sign the certificate.
- Cipher suites selected.
- Group key shared (ECDHE).
- Signature algorithm (RSA).
- Server certificate.
- Verify the certificate (RSA).
- Generate the master key from the public server and client private key (HMAC-SHA2)
- Master key.
- Client ready.
- Decrypts the Master key.
- Server ready.
- Data exchange.
- Data exchange.
- Data exchange.
Data exchange using AES-GCM, AES-CCM or ChaCha20-Poly1305.

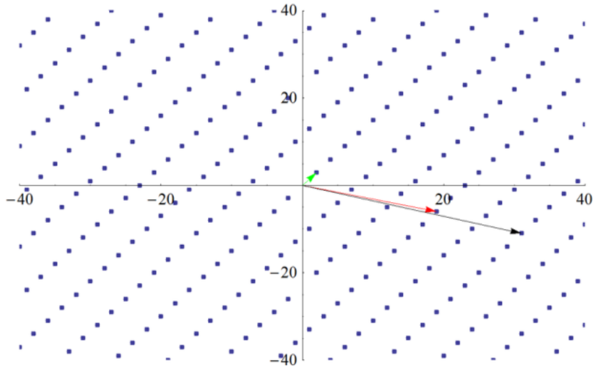
TLS 1.3 Protocol

NIST PQC Process

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

CRYPTO QUANTIQUE

- Announced in December 2016
- 82 candidate algorithm were submitted
- 3 rounds of evaluation
- 1 Key-encapsulation (KEM) and 3 digital signature schemes were selected



Candidates selected:

- CRYSTALS-KYBER: (M) Lattice-based public-key encapsulation
- CRYSTALS-Dilithium: (M) Lattice-based digital signature scheme
- FALCON: Lattice-based digital signature
- SPHINCS+: Hash-based digital signature

FIPS Standards

- FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard
- FIPS 204, Module-Lattice-Based Digital Signature Standard
- FIPS 205, Stateless Hash-Based Digital Signature Standard

- Standard based on FALCON* will be published later

Fast Fourier Lattice-based Compact Signatures over NTRU
*N-th degree Truncated Polynomial Ring Units

Supporting PQC algorithms in MCUs

Example PQC Algo execution times:

- Target MCU is ST Microelectronics Cortex-M7 (216MHz)
- M7 has a 64-bit FPU (M4 has 32-bit)
- Falcon requires 53-bit floating point precision

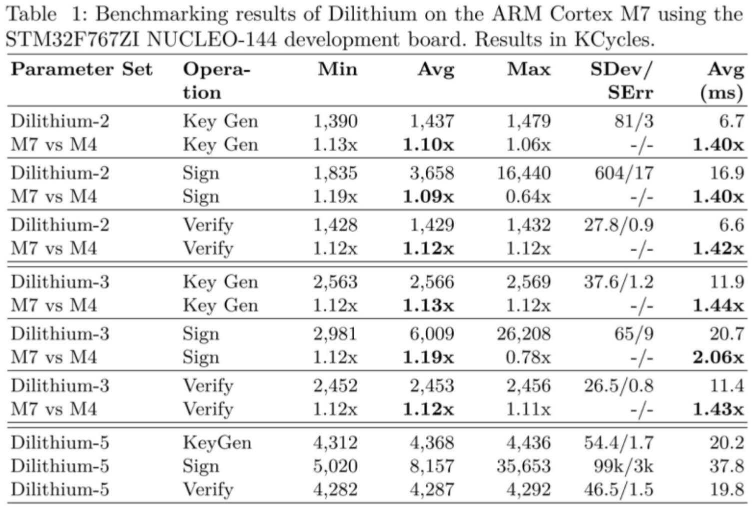


Table 1: Benchmarking results of Dilithium on the ARM Cortex M7 using the STM32F767ZI NUCLEO-144 development board. Results in KCycles.

Parameter Set	Operation	Min	Avg	Max	SDev/SErr	Avg (ms)
Dilithium-2	Key Gen	1,390	1,437	1,479	81/3	6.7
M7 vs M4	Key Gen	1.13x	**1.10x**	1.06x	-/-	**1.40x**
Dilithium-2	Sign	1,835	3,658	16,440	604/17	16.9
M7 vs M4	Sign	1.19x	**1.09x**	0.64x	-/-	**1.40x**
Dilithium-2	Verify	1,428	1,429	1,432	27.8/0.9	6.6
M7 vs M4	Verify	1.12x	**1.12x**	1.12x	-/-	**1.42x**
Dilithium-3	Key Gen	2,563	2,566	2,569	37.6/1.2	11.9
M7 vs M4	Key Gen	1.12x	**1.13x**	1.12x	-/-	**1.44x**
Dilithium-3	Sign	2,981	6,009	26,208	65/9	20.7
M7 vs M4	Sign	1.12x	**1.19x**	0.78x	-/-	**2.06x**
Dilithium-3	Verify	2,452	2,453	2,456	26.5/0.8	11.4
M7 vs M4	Verify	1.12x	**1.12x**	1.11x	-/-	**1.43x**
Dilithium-5	KeyGen	4,312	4,368	4,436	54.4/1.7	20.2
Dilithium-5	Sign	5,020	8,157	35,653	99k/3k	37.8
Dilithium-5	Verify	4,282	4,287	4,292	46.5/1.5	19.8

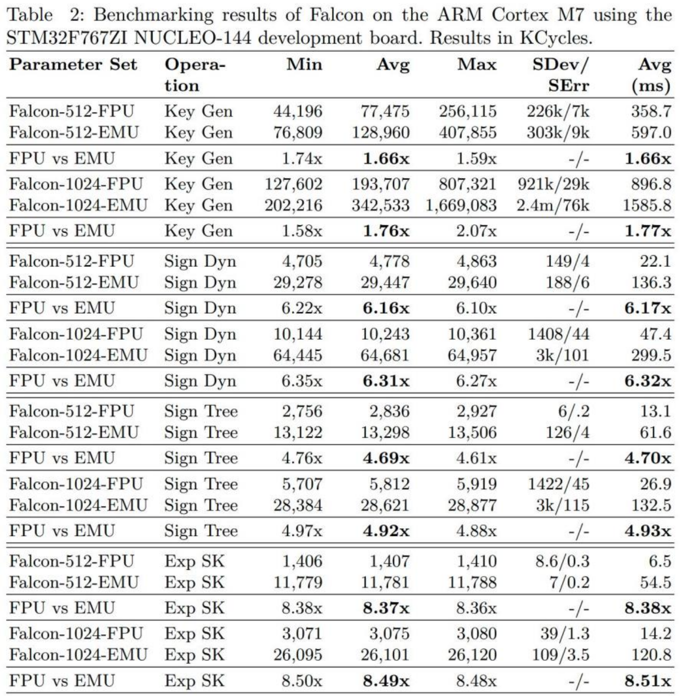


Table 2: Benchmarking results of Falcon on the ARM Cortex M7 using the STM32F767ZI NUCLEO-144 development board. Results in KCycles.

Parameter Set	Operation	Min	Avg	Max	SDev/SErr	Avg (ms)
Falcon-512-FPU	Key Gen	44,196	77,475	256,115	226k/7k	358.7
Falcon-512-EMU	Key Gen	76,809	128,960	407,855	303k/9k	597.0
FPU vs EMU	Key Gen	1.74x	**1.66x**	1.59x	-/-	**1.66x**
Falcon-1024-FPU	Key Gen	127,602	193,707	807,321	921k/29k	896.8
Falcon-1024-EMU	Key Gen	202,216	342,533	1,669,083	2.4m/76k	1585.8
FPU vs EMU	Key Gen	1.58x	**1.76x**	2.07x	-/-	**1.77x**
Falcon-512-FPU	Sign Dyn	4,705	4,778	4,863	149/4	22.1
Falcon-512-EMU	Sign Dyn	29,278	29,447	29,640	188/6	136.3
FPU vs EMU	Sign Dyn	6.22x	**6.16x**	6.10x	-/-	**6.17x**
Falcon-1024-FPU	Sign Dyn	10,144	10,243	10,361	1408/44	47.4
Falcon-1024-EMU	Sign Dyn	64,445	64,681	64,957	3k/101	299.5
FPU vs EMU	Sign Dyn	6.35x	**6.31x**	6.27x	-/-	**6.32x**
Falcon-512-FPU	Sign Tree	2,756	2,836	2,927	6/.2	13.1
Falcon-512-EMU	Sign Tree	13,122	13,298	13,506	126/4	61.6
FPU vs EMU	Sign Tree	4.76x	**4.69x**	4.61x	-/-	**4.70x**
Falcon-1024-FPU	Sign Tree	5,707	5,812	5,919	1422/45	26.9
Falcon-1024-EMU	Sign Tree	28,384	28,621	28,877	3k/115	132.5
FPU vs EMU	Sign Tree	4.97x	**4.92x**	4.88x	-/-	**4.93x**
Falcon-512-FPU	Exp SK	1,406	1,407	1,410	8.6/0.3	6.5
Falcon-512-EMU	Exp SK	11,779	11,781	11,788	7/0.2	54.5
FPU vs EMU	Exp SK	8.38x	**8.37x**	8.36x	-/-	**8.38x**
Falcon-1024-FPU	Exp SK	3,071	3,075	3,080	39/1.3	14.2
Falcon-1024-EMU	Exp SK	26,095	26,101	26,120	109/3.5	120.8
FPU vs EMU	Exp SK	8.50x	**8.49x**	8.48x	-/-	**8.51x**

https://csrc.nist.gov/csrc/media/Presentations/2022/benchmarking-and-analysing-nist-pqc-lattice-based/images-media/session4-howe-benchmarking-analysing-pqc2022.pdf

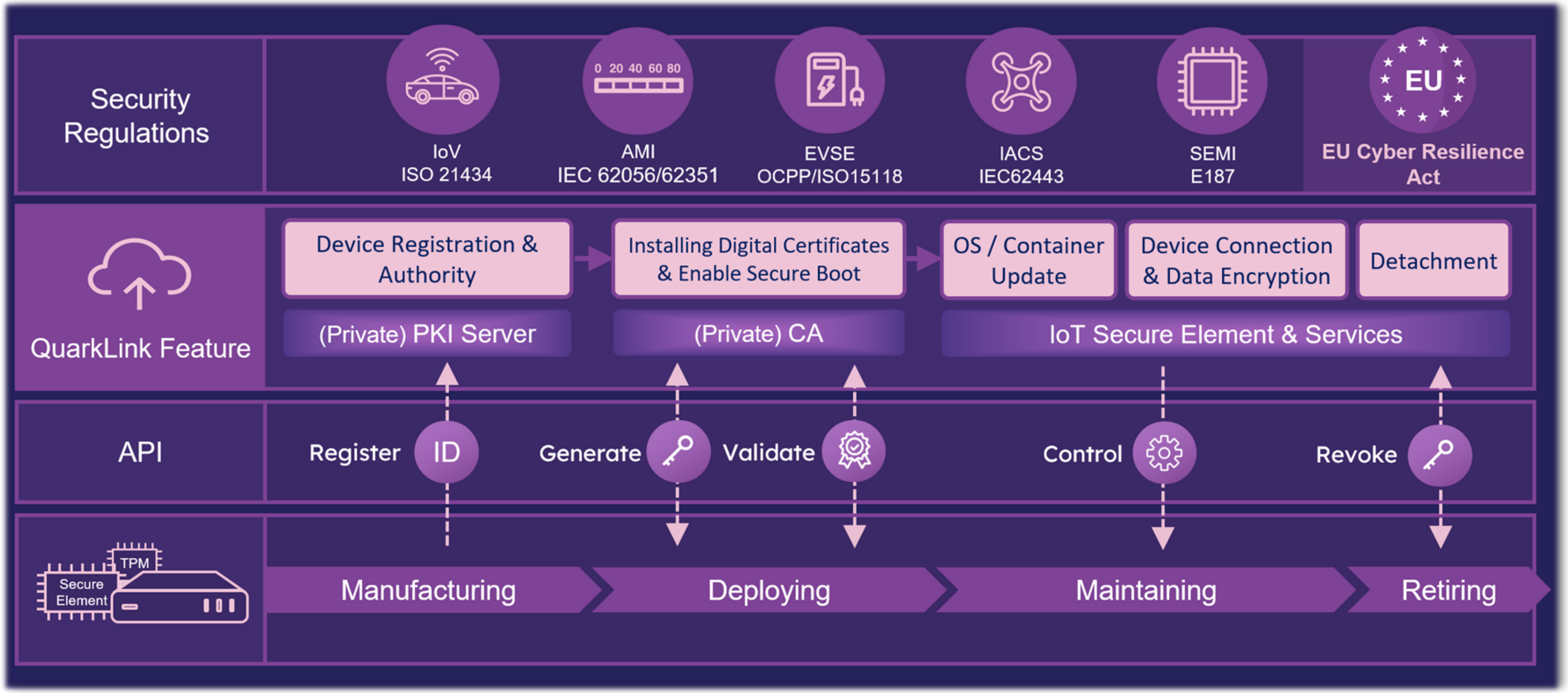
How are keys and certificates managed?

Typically, keys and certificates are generated on local PC/Laptops using **OpenSSL functions**:

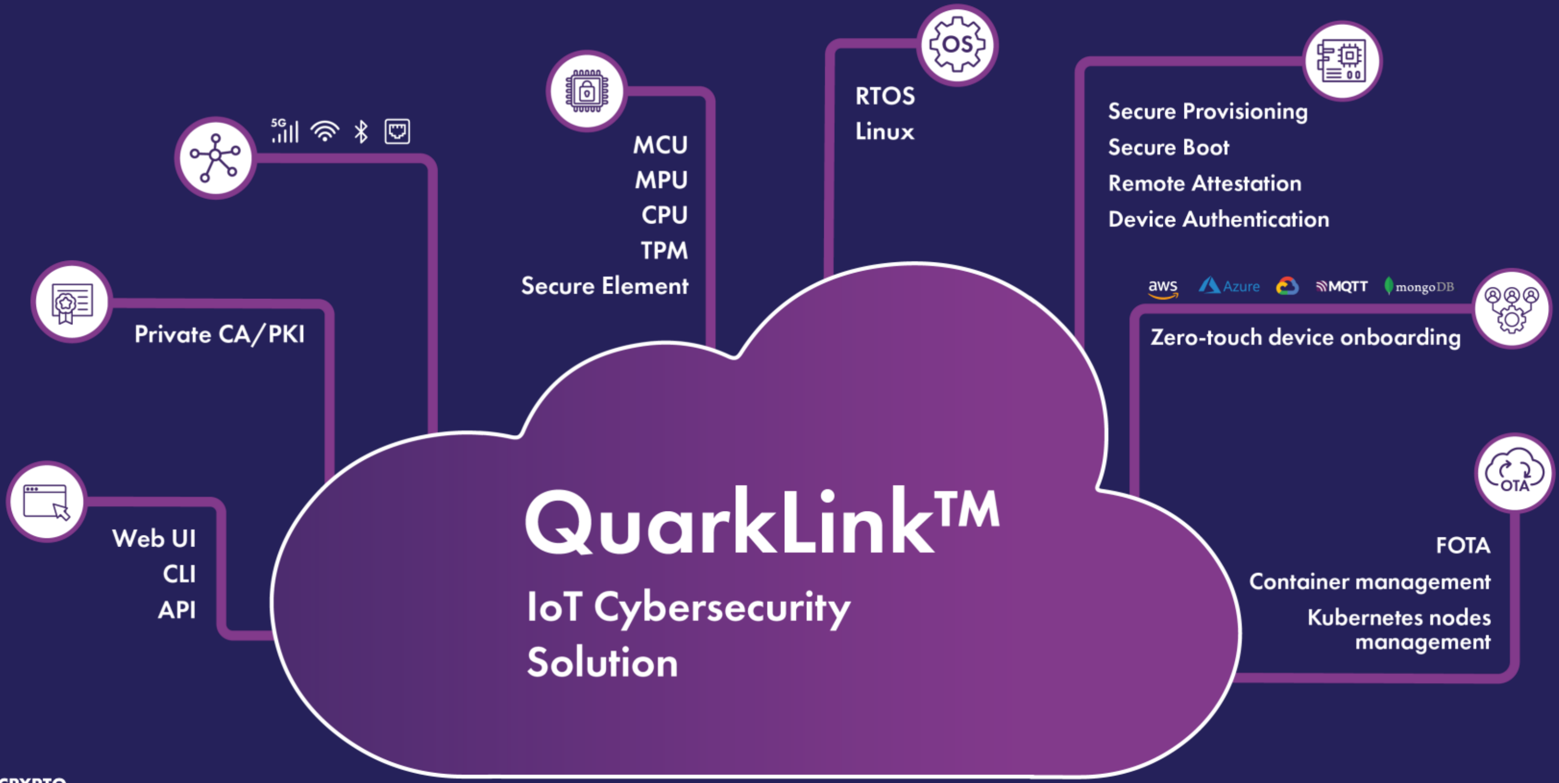
- Generate RSA or ECC key pair (where are the private keys saved?)
- Generate random number
- Generate signature digest
- Verify signature digest
- Encrypt/Decrypt using asymmetric keys
- Encrypt/Decrypt using symmetric keys
- Generate cryptographic HASH
- TLS handshake functions
- Generate certificate
- Verify X.509 signature
- Derive shared secret using ECDH

➢ Support for Post Quantum Algorithms is evolving (OpenSSL 3.0)
➢ Open Quantum Safe Project (OQS) (fork of OpenSSL)

Device Life-Cycle Management



QuarkLink addresses the security needs for edge devices at the embedded, OS and the cloud



CRYPTO QUANTIQUE

Securing the connected world with zero trust

Summary

CRYPTO QUANTIQUE

PQC Adoption Challenges

- Validation & Testing
  - CAVP and CMVP are providing test vectors for FIPS 140 certification
- Transition period
  - Interoperability and performance e.g.; TLS, SSH, HSM
  - IETF drafts to RFC documents
- Support for PQC with current System-On-Chip devices
  - Hybrid scheme
    - Use of pre-quantum scheme and a PQC scheme in parallel
    - Integration of PQC schemes into pre-quantum scheme
- New side channel attacks
- Requirement for additional general-purpose signature schemes that are not based on lattices
  - Smaller public keys and signatures
  - New NIST initiatives on going since 2023 (Crypto Diversity)

Demo time!

Thank you.