CRYPTO QUANTIQUE

CQ Initial Enrolment Firmware

Version V1.10

Table of Contents

1 Scope

This document is targeted at embedded software engineers who require additional details regarding the design of the Initial Enrolment Firmware (IEF) that is integrated into the CQ QuarkLink security platform. The IEF is a software package that is proprietary to Crypto Quantique and is provided to users as a compiled image that is provisioned into target IoT devices during the ***Provisioning*** function of a QuarkLink (see QuarkLink User Guide). The IEF is closely linked to the secure bootloader that is initially provisioned into a target IoT device. The IEF is specific to the target hardware being provisioned.

2 Reference Material

The following additional reference material is recommended for review:

- QuarkLink User Guide (visit our GitHub)
- CQ Bootloaders (visit our GitHub)
- QuarkLink Demonstration Videos (https://www.youtube.com/@CryptoQuantique)
- Crypto Quantique GitHub (https://github.com/cryptoquantique)

3 Target Hardware

The QuarkLink supports multiple hardware devices, typically secure microcontrollers from silicon manufacturers such as Espressif, Renesas, STMicroelectronics, NXP etc. The secure microcontrollers are used to build IoT devices which can be onboarded to secure cloud services using a QuarkLink.

Crypto Quantique provides QuarkLink users with embedded firmware to ensure that they (the users) can build secure IoT products that can be managed, throughout their life-cycle, by including capabilities such as firmware Over-the-Air (OTA) updates, certificate renewal, certificate revocation and onboarding to cloud service providers, MQTT brokers and databases.

Each target hardware is supported by a secure bootloader and an IEF instance. The secure bootloaders are available in both binary and source (via GitHub) whereas the IEF is binary only (developed by Crypto Quantique to ensure security).

The QuarkLink is preconfigured with all bootloaders and IEF instances required to provision all supported target hardware.

Each target hardware supported by QuarkLink has unique customisations which are dependant of the design of the security hardware in the device. This document includes appendices that cover the differences for each device.

4 IEF Architecture

The IEF is an application that is executed on the target hardware during the provisioning process and carries out the following functions:

- Configures a UART to allow communication with the QuarkLink provisioning tool
- Executes the Crypto Quantique QuarkLink client library
- Connects the target hardware to the local WiFi network (to communicate with the QuarkLink instance)
- Generates the target hardware identity cryptographic keys and stores them securely
- Configures the QuarkLink batch with the target hardware device ID.
- Checks the QuarkLink instance for firmware updates
- Executes firmware update process

4.1 IEF Security

The IEF is a proprietary firmware image that is generated by Crypto Quantique. The IEF image is signed by the QuarkLink instance that is carrying out the provisioning task. During provisioning, a secure bootloader is flashed into the target hardware first, followed by the IEF. On a reset the bootloader verifies the signature of the IEF image prior to executing it (see Appendix I). Effectively, the bootloader is a second stage bootloader and the IEF represents the user application of the target hardware.
The memory map of the ESP32 device once it has been provisioned is shown in Figure 1.

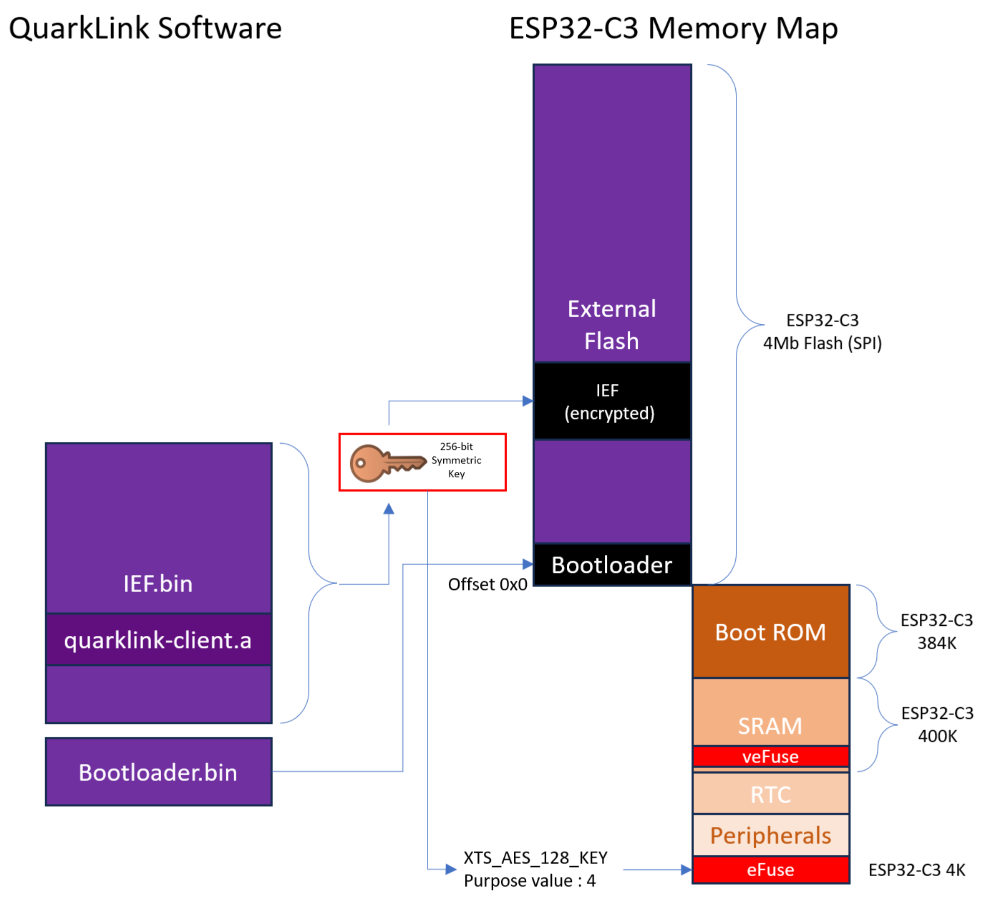


Figure 1: ESP32 Memory Map post QuarkLink provisioning

Once the bootloader has verified the IEF image it will execute the IEF. Figure 2 is a flow diagram showing the execution steps of the IEF. Let's examine in detail to function of each step.

4.1.1 Initialise UART

The target hardware is required to communicate with the Laptop/PC via a USB cable. This enables communication between the target hardware and the QuarkLink UI which is running in a browser. Typically the target hardware includes a UART-USB converter silicon chip which manages the UART configuration.

4.1.2 Initialise WiFi Module

The target hardware checks its configuration memory to determine whether the WiFi parameters have been pre-initialised. If they have not, the target hardware requests the QuarkLink to prompt the user to input the WiFi details. Once input by the user, the local WiFi SSID and Password are sent to the target hardware, over the UART, and programmed into the configuration memory.

4.1.3 QuarkLink Instance Parameters

The target hardware checks its configuration memory to determine whether the QuarkLink parameters have been pre-initialised. If not, the target hardware makes requests of the QuarkLink over the UART. The type of requests are shown below:

• ql_endpoint:	request the QuarkLink instance URL (e.g. *myql.quarklink.io*)
• ql_port:	request the QuarkLink instance port (default 6000)
• ql_rootCA:	request the QuarkLink instance root certificate (base64-encoded)

4.1.4 QuarkLink Initialisation

During this next step, the quarklink-client library is invoked to generate and store cryptographic keys to represent the identity of the target hardware. A pair of asymmetric keys are generated, called the enrolment key pair (ESK, EPK). The enrolment key pair are used to prove the unique identity of the target hardware (***DeviceID***) during the enrolment onto the QuarkLink.

Note: ESK – Enrolment Secret Key, EPK – Enrolment Public Key

An additional asymmetric key pair is generated during this initialisation phase **IF** the target hardware does not support runtime key generation i.e. the target hardware does not include cryptographic capability to generate asymmetric keys on-chip.

If the target device **does** include hardware support for cryptographic key generation during runtime, the second key pair is **only** generated during the enrolment phase when a device has been recognised as a valid device that is present in a Batch. This second key pair is called the Device Key pair (DSK, DPK). This key pair is unique and used for connection to the cloud service (AWS, Azure etc) as specified in the security policy associated to the batch.

Note: DSK – Device Secret Key, DPK – Device Public Key

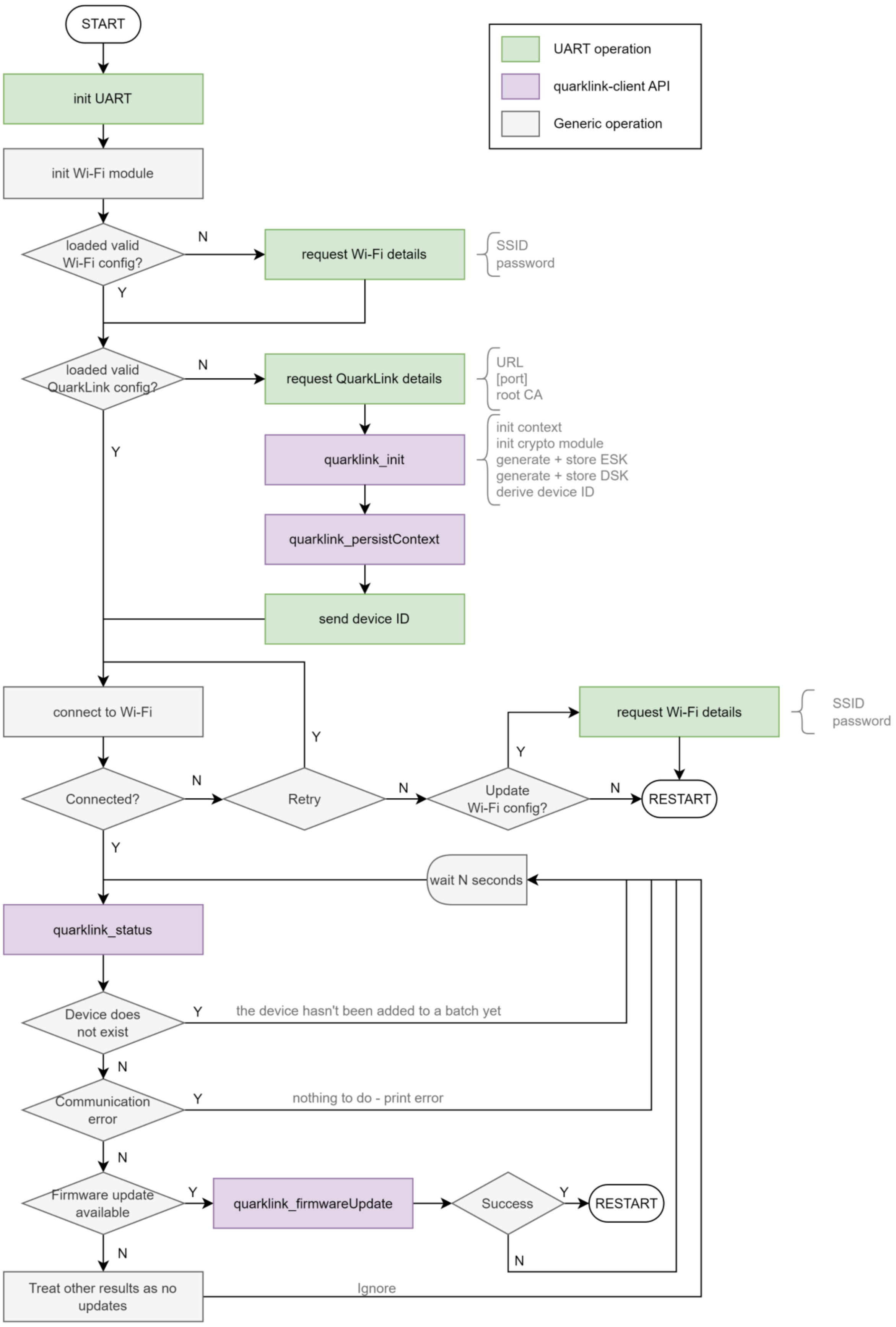
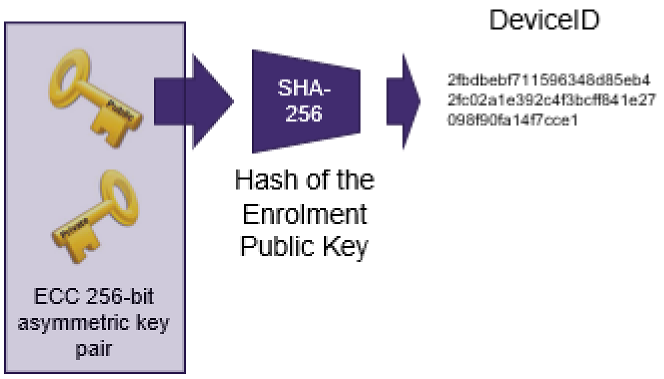


Figure 2: IEF flow

4.1.5 Send DeviceID

Once the enrolment key pair have been generated, the quarklink-client library generates a SHA-256 HASH of the enrolment public key (EPK). This 256-bit number represents the unique identity of the target hardware and is called the ***DeviceID***. The target hardware sends the ***DeviceID*** to the QuarkLink (the QuarkLink will copy this value into the ***Batch*** defined in the policy).


DeviceID
Public
Private
ECC 256-bit asymmetric key pair
SHA-256
Hash of the Enrolment Public Key
2fbdbebf711596348d85eb4
2fc02a1e392c4f3bcff841e27
098f90fa14f7cce1

4.1.6 Connect to WiFi

The target hardware will connect to the local WiFi network and setup a WiFi communications link with the QuarkLink instance. This means that there are now two communication links to the QuarkLink (UART & WiFi). The WiFi link is created because this provides a secure TLS connection to the QuarkLink allowing secure communication between target hardware and QuarkLink.

4.1.7 QuarkLink Status

The target hardware will execute an API call to the QuarkLink instance over the secure WiFi communications channel to request the status. The QuarkLink will update the target hardware which will then carry out several checks based on the status (see Figure 2 & below):

4.1.7.1 Device in a Batch

Is the ***DeviceID,*** that was previously sent to the QuarkLink, included in a Batch? This is a check that the QuarkLink has been configured to allow this particular device to enrol onto the QuarkLink and receive security credentials.

4.1.7.2 QuarkLink Firmware Update

The QuarkLink status has indicated that a new firmware version is available for download from the QuarkLink. The firmware update process will now start by calling the appropriate quarklink-client library API. On completion of the firmware update having been flashed into the device, the device will reset.

Note: Once the WiFi credentials and QuarkLink configuration i.e. URL and Root certificate, have been programmed into the configuration Flash memory (see 4.1.2), the target hardware can be disconnected from the UART. Firmware updates for the device will then be checked via QuarkLink status request over the WiFi, no further UART connection is necessary.

5 Appendix I – Espressif ESP32

ESP32 is a Wi-Fi and Bluetooth 5 (LE) System-On-Chip (SoC). It is classed as a secure MCU due to its integration of secure peripherals and functions such as Secure Boot. The ESP32 has experienced issues with older devices whereby the security features have been bypassed by the use of fault injection techniques. These issues have been resolved in newer versions and are now supported by QuarkLink. QuarkLink utilises the Secure Boot V2 (ECO 3 onwards) scheme.

5.1 Secure Boot V2

Secure Boot protects a device from running any unauthorized (i.e., unsigned) code by checking that each piece of software that is being booted is signed. On an ESP32, these pieces of software include the second stage bootloader and each application binary. Note that the first stage bootloader does not require signing as it is ROM code thus cannot be changed.

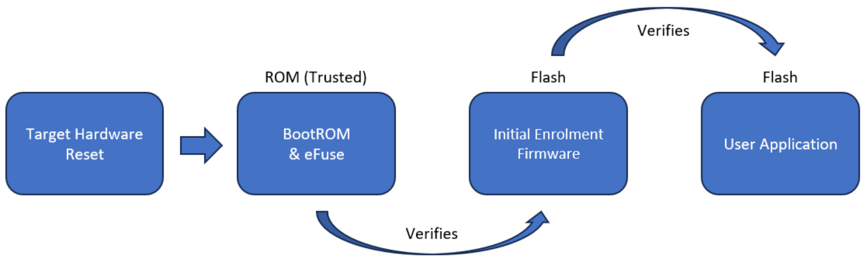
A RSA based Secure Boot verification scheme (Secure Boot V2) is implemented on ESP32 (ECO 3 onwards).

The Secure Boot process on the ESP32 involves the following steps:

1. When the first stage bootloader loads the second stage bootloader (IEF), the IEFs RSA-PSS (**RSA**-**P**robabilistic **S**ignature **S**cheme) signature is verified. If the verification is successful, the IEF is executed.

2. When the IEF loads a particular user application image, the application's RSA-PSS signature is verified. If the verification is successful, the user application image is executed.

The diagram below shows the process, often referred to as the chain of trust.



The advantages of this scheme are as follows:

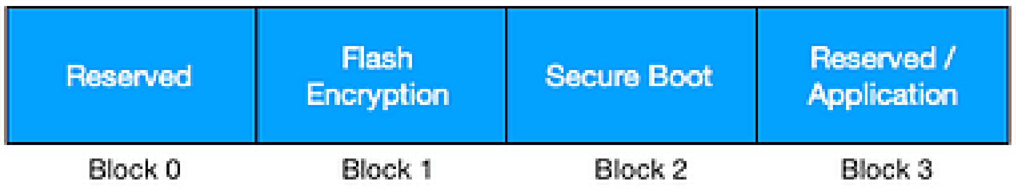
- The RSA-PSS public key is stored on the device. The corresponding RSA-PSS private key is kept within the Hardware Security Module (HSM) of the QuarkLink and is never accessed by the device.
- Only one public key can be generated and stored in the chip during QuarkLink provisioning.
- Same image format and signature verification method is applied for user applications and IEF.
- No secrets are stored on the device. Therefore, it is immune to passive side-channel attacks (timing or power analysis, etc.)

For more detailed information on the secure boot process visit the Espressif website:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/secure-boot-v2.html

5.2 eFuse

The eFuse plays an important role in the functioning of the ESP32 security features. The ESP32 has a 1024-bit eFuse, which is a one-time programmable memory. This eFuse is divided into 4 blocks of 256-bits each.



ESP32 eFuse Overview

Of primary interest to QuarkLink users are blocks 1 and 2. These blocks store keys for flash encryption and secure boot respectively. Also, once the keys are stored in the eFuse, the IEF configures it in such a way that any software running on ESP32 cannot read (or update) these keys (**disable software readout**). Once **disable software readout** has been enabled by the IEF, only the ESP32 hardware can read and use these keys for ensuring secure boot and flash encryption.

Important Note: Once the Secure Boot eFuse has been blown, the target hardware cannot be updated with firmware unless it has been correctly signed by the associated QuarkLink instance. Also, the target hardware cannot be erased to restore it to the factory default condition, it is non-recoverable. Please ensure that when provisioning via the QuarkLink the production/release version of the bootloader is not provisioned unless the user is fully conversant with the implications of enabling maximum security for their target hardware.

5.2.1 veFuse

Once a key is provisioned into eFuse (see section 5.2), it cannot be erased or changed once the Secure Boot eFuse has been blown. This is required for security reasons when the eFuse is used to store secret keys needed to verify firmware images prior to execution. This feature can, however, be inconvenient during protoyping or unit testing of a connected device as any temporary keys, that will not to be used during production (release), cannot be erased, thus making the device unusable and unrecoverable.

To help users during prototyping, Crypto Quantique have utilised the **Virtual eFuse** or veFuse. This is a software modification that allows virtual keys to be programmed into SRAM memory and therefore, modified or erased at any time. During startup, the eFuses are copied to SRAM. All eFuse operations (read and write) are performed with SRAM instead of the real eFuse registers.This allows users to fully test their connected device with temporary cryptographic keys. Once all unit tests and prototyping are complete, the user firmware can be rebuilt using the correct configuration.

The default build environment is ***esp32-c3-vefuses***. To build the firmware with the ***esp32-c3-release*** configuration, run the command :

```
>pio run -e esp32-c3-release
```

5.3 Bootloader

As part of the provisioning task, the QuarkLink programs the ESP32 with a secure bootloader. During the programming phase, the Public key of the signing key selected as part of the provisioning task is programmed into the eFuse area of the target hardware (see section 5.2) as shown in Figure 3.

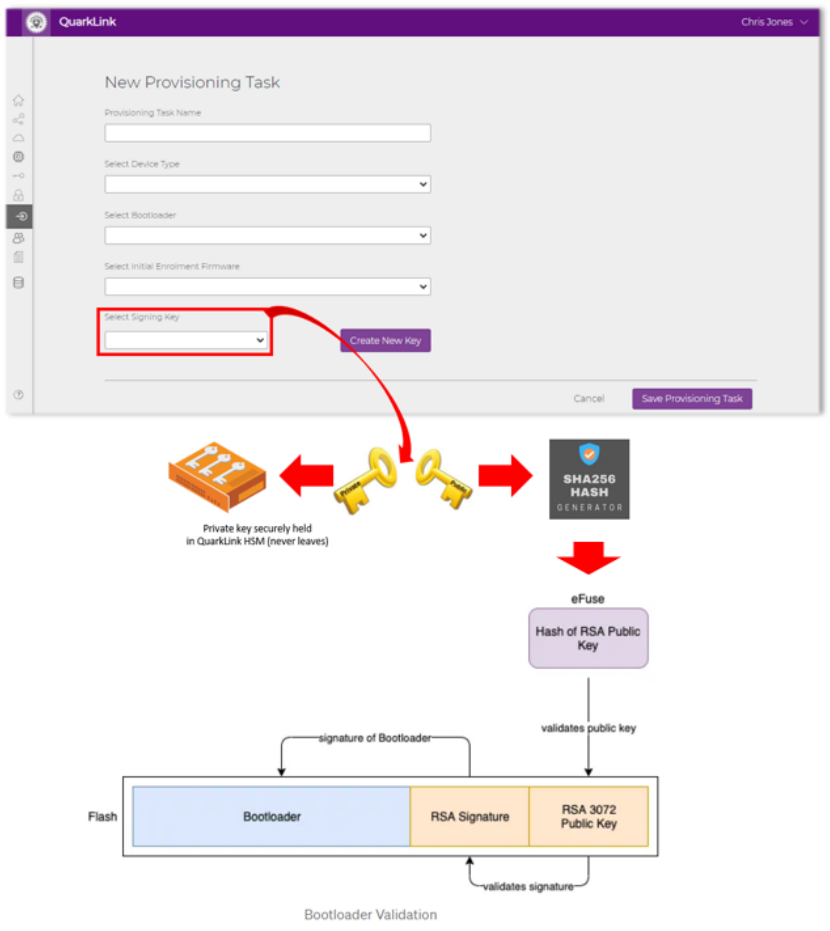


Figure 3: Secure boot signing key configuration

- **Bootloader Image:** This is the bootloader executable that is selected by the QuarkLink user during ***Provisioning***.
- **RSA Signature:** This is the RSA3072 based signature of the bootloader image. The key is generated by QuarkLink and is created/selected during the creation of a ***Provisioning Task***.
- **RSA 3072 Public Key:** The public key that can be used to validate the signature.

For more information regarding the secure bootloader implemented in the QuarkLink ESP32 provisioning process, please review document ***CQ Bootloaders*** application note.

5.4 Security Hardware

As described in this document, the IEF is an application that is executed on the target hardware during the provisioning process and carries out several functions one of which is the generation and storage of cryptographic keys. With the ESP32 several approaches can be used for this important security function, they are;

(1) use the ESP32 Digital Signature (DS) Peripheral,
(2) use a secure element such as the ATECC608, or
(3) store the keys in an encrypted NVS flash partition.

5.4.1 ESP32 Digital Signature Peripheral

Most recent ESP32 devices, including ESP32-C3 and ESP32-S3, are provided with the DS peripheral. Using this module is the preferred solution as it ensures the highest level of security.

With this approach, private keys are handled directly within the DS hardware and are not accessible by software in any way. Similarly to Secure Boot and Flash Encryption, keys are secure by means of eFuses.

Important note: the DS hardware is directly connected to the eFuse registers and cannot access the RAM: this means it is not compatible with virtual eFuses and will always use real eFuses.

At first boot, QuarkLink initialises the DS peripheral with two keys (see 4.1.4) and the information related to these keys is stored in encrypted Flash. The user can therefore re-provision the same device as many times as they want, as long as the Flash is not fully erased. In the event of the Flash being fully erased, the DS-related data will be lost and two new keys will be generated as part of the initialisation process; because real eFuses are used and because of the limited availability of eFuses this process can only be repeated **3** times, after which the device will run out of available eFuses and fail to initialise.

The user needs to be mindful of this and avoid erasing the Flash.
It is recommended that the user familiarise themselves with the operation of the DS peripheral and associated eFuses. For more in-depth information on the Digital Signature peripheral visit https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-reference/peripherals/ds.html

For more details on eFuses visit https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-reference/system/efuse.html

5.4.2 ATECC608

The M5Stack EduKit module includes an ATECC608 secure element, and this will be used for secure key generation and storage when selecting one of the *ief-m5edukit-ecc608* IEFs when provisioning the device. The QuarkLink utilises key slot number 0 in the ATECC608 to store cryptographic keys required for secure onboarding.

For more details on the M5Stack EduKit visit https://shop.m5stack.com/products/m5stack-core2-esp32-iot-development-kit-for-aws-iot-edukit

For more details on the Microchip ATECC608 visit https://www.microchip.com/en-us/product/ATECC608B-TNGTLS

5.4.3 NVS Partition

When neither the DS Peripheral nor a secure element are available, the cryptographic keys are stored in an encrypted NVS partition. This means that the keys are encrypted at rest and cannot be compromised by directly reading from the flash chip, however the keys are in memory in plain text at runtime so this approach is not as secure as when using the DS Peripheral or secure element approaches.

More details on NVS encryption in the ESP32 can be found at https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_encryption.html

To have a view of the partition table structure used by QuarkLink's IEF and ESP32 examples visit our GitHub page quarklink-getting-started-esp32-platformio/partition-tables at main · cryptoquantique/quarklink-getting-started-esp32-platformio (github.com)

6 Revision History

CQ Initial Enrolment Firmware

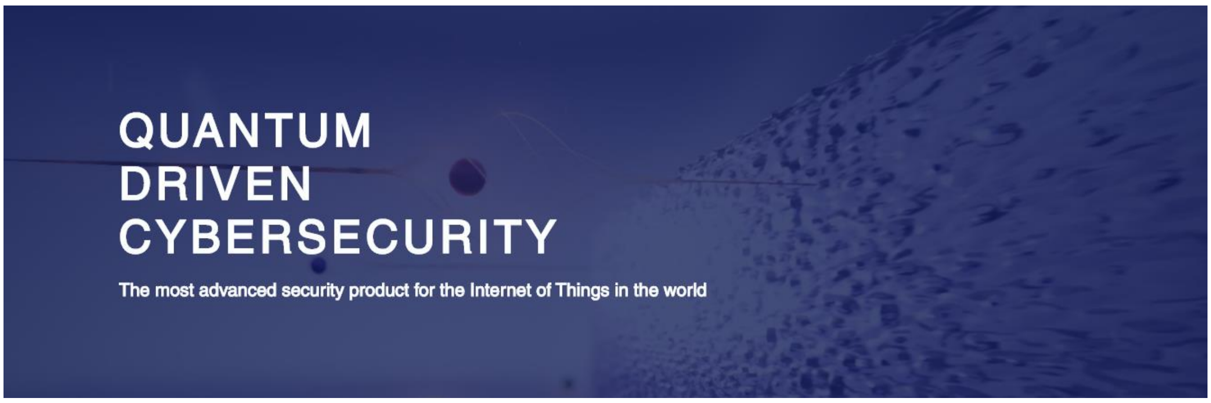
Rev.	Date	Owner	Description
1.00	21.8.2023	CDJ	Original document
1.01	18.9.2023	CDJ	Update with new veFuse function
1.02	22.9.2023	AE/AM	Addition of DS peripheral and NVS explanation
1.10	13.10.2023	AE	Additional technical information regarding the DS peripheral

CRYPTO
QUANTIQUE

United Kingdom

Unit 304-5,
164-180 Union Street,
London
SE1 0LH

General contact email:
info@cryptoquantique.com


QUANTUM
DRIVEN
CYBERSECURITY
The most advanced security product for the Internet of Things in the world