

ARTIK Security

Wei Xiao

Oct 23rd, 2018



Agenda

- ARTIK Module Security
- ARTIK Platform Security
- S-Module Design Considerations
- Security Use Case and Demo

IoT hacking and Vulnerabilities



Hackable Cardiac Devices from St.Jude

Hackers can access St. Jude's implantable cardiac devices, deplete the battery or administer incorrect pacing or shocks

Mirai DDoS

IoT devices with weak password being used as bots to launch DDoS attack

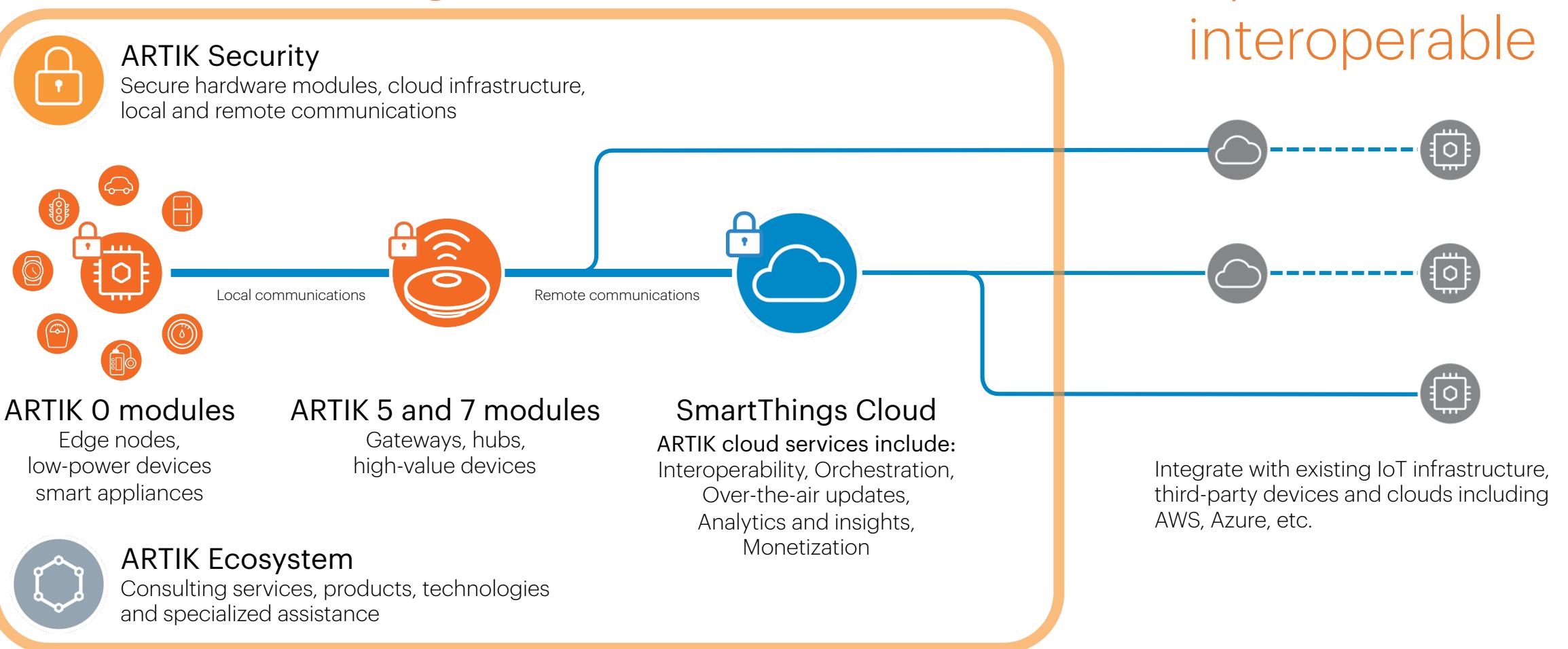


Source: <https://www.iotforall.com/>

Samsung ARTIK™ IoT Platform

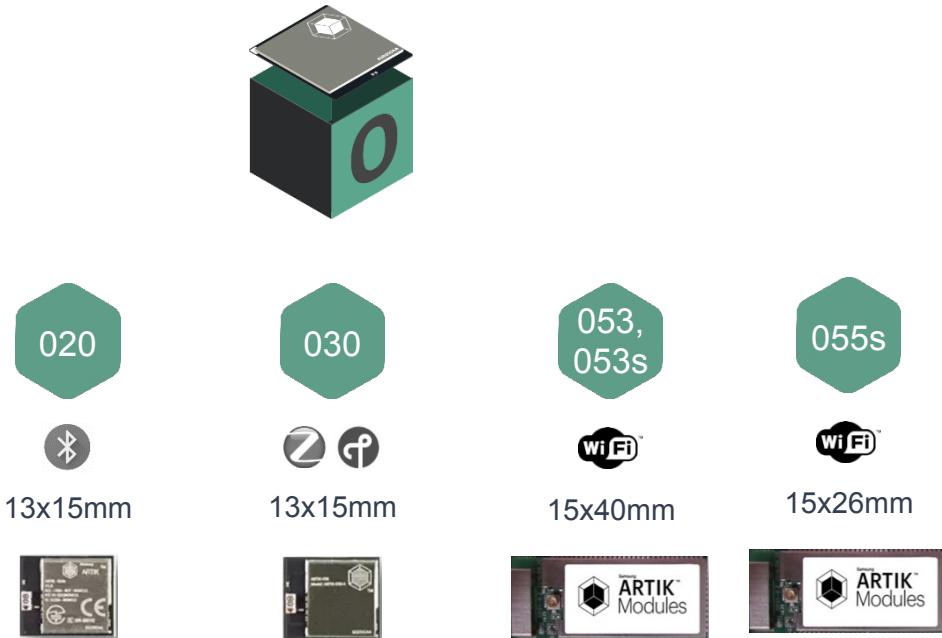
End-to-end integration...

...Open and
interoperable



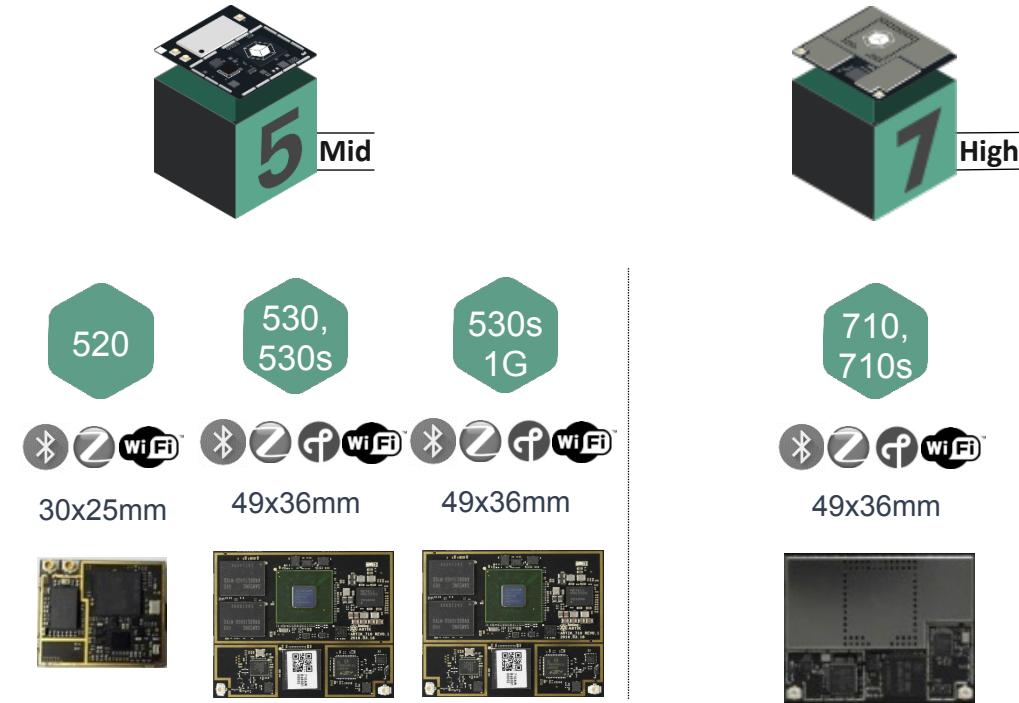
ARTIK Modules Portfolio

Battery-powered Devices, Edge Nodes, Sensors and Actuators



- Cortex-M, Cortex-R CPUs
- Single, Dual protocol radios
- Internal RAM/Flash

Hubs, Gateways



- Single, Dual, Quad, Octa core Cortex-A CPUs
- Multiple protocol radios
- Linux based platforms

ARTIK Module Security



Regular vs. S Modules

- Same HW specifications!
- "S" type modules can be identified by **blue** labeling

Standard module



"S" module



Security Questionnaire

How do you provide security across all attack surfaces?

Question	ARTIK 5/7/053
Do you support secure communication from device to device or device to cloud? How do you secure communication? Are you using TLS1.2 or higher?	Yes, HTTPs using TLS 1.2
How do you establish identity of device?	Using unique certificate on each device
How does the device establish identity of cloud?	Both device and cloud are chained to ARTIK Root CA and can verify each other certificates
Do you have mutual authentication when enabling secure communication?	Yes
Do you have the infrastructure to inject unique key and certificate in each device to establish unique identity per device? How much does it cost?	Yes (Done at Samsung factory. Cost included in module)
How do you protect your certificate, keys?	Specialized HW on module (secure element)
Is your certificate infrastructure secure? How do you secure your Root Certificates? How much does it cost?	Yes (Root CA secured by 3 rd party security vendor)
How do you guarantee your firmware integrity?	Secure Boot

Samsung ARTIK™ S-Module Features

	ARTIK module (530, 710)	ARTIK S-module (530s/710s)	ARTIK module (053)	ARTIK S-Module (053s/055s)	Comments	
Secure communication	Per device unique key & cert	✓	✓	✓	✓	Uniquely identifies device
	Key stored in HW secure storage (secure element or secure key storage)	✓	✓	✓	✓	Secure key storage
	PKI infrastructure: Mutual authentication of device and cloud	✓	✓	✓	✓	Device communicates to authorized cloud and vice versa
	Post Provisioning		✓		✓	Provision with your own keys and certificates
Device protection/ secure code execution	KMS infrastructure for code signing		✓		✓	Key Management Service
	Code verification key in HW		✓		✓	Secure key storage
	Secure boot (verified boot on 530s/710s)		✓		✓	Boot image verification
	JTAG access locked		✓		✓	Lock out debug access
Data protection/ Secure storage	Secure OS (separate normal & secure operations)		✓			Hardware enforced secure applications via TEE
	Security Lib API (27 API calls)	get_cert(), gen_rand(), get_sig()	✓	get_cert(), gen_rand(), get_sig()	✓	Key Manager, Authentication, Secure Storage, Post Provisioning, Encrypt/Decrypt
	Secure storage		✓		✓	Secure Element & Encrypt data stored on Flash

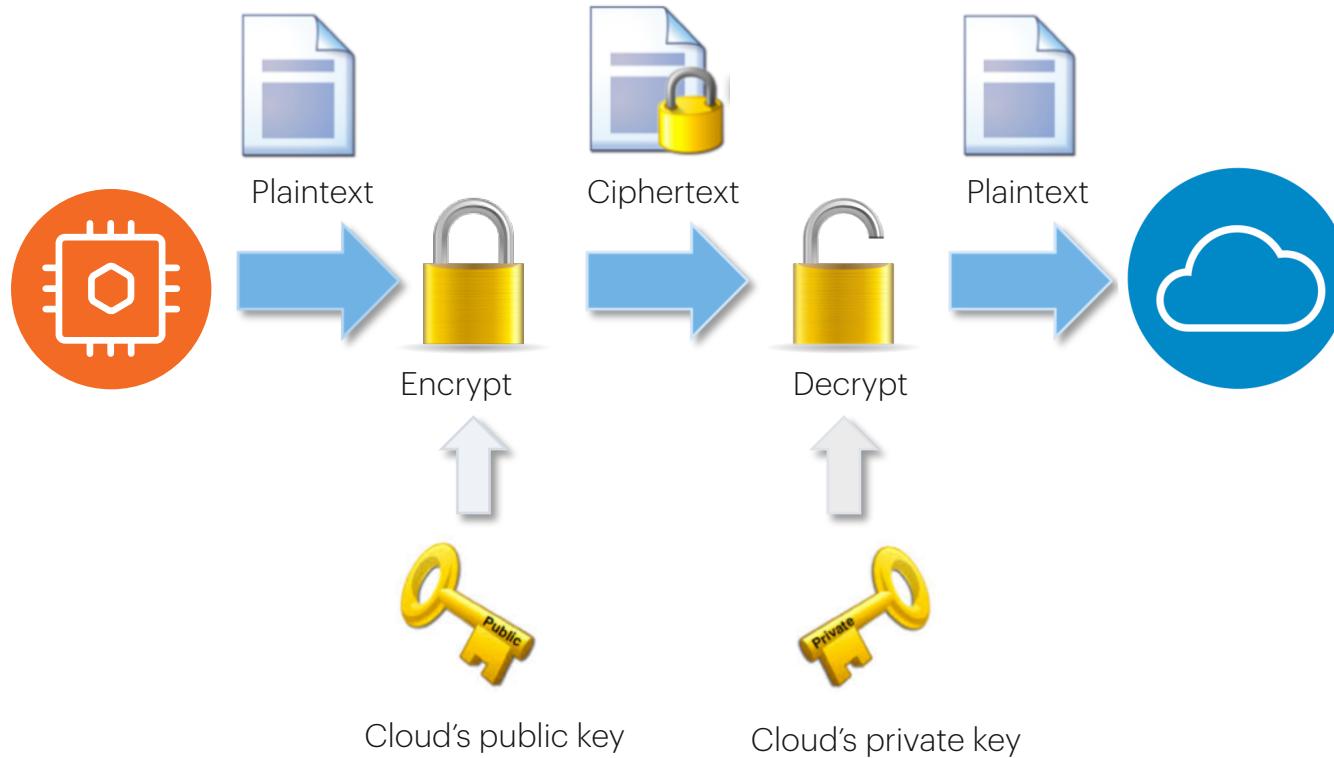


ARTIK™

Secure storage

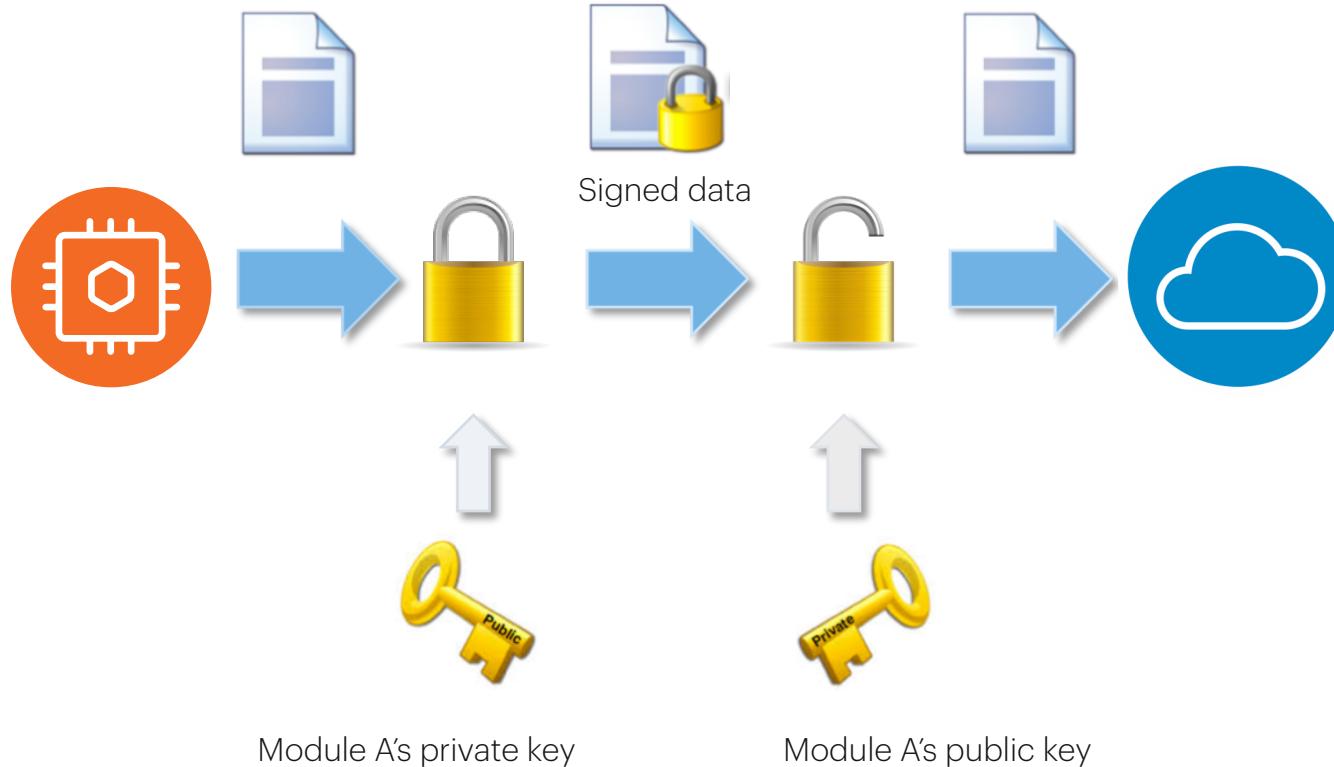
Confidential

Encryption



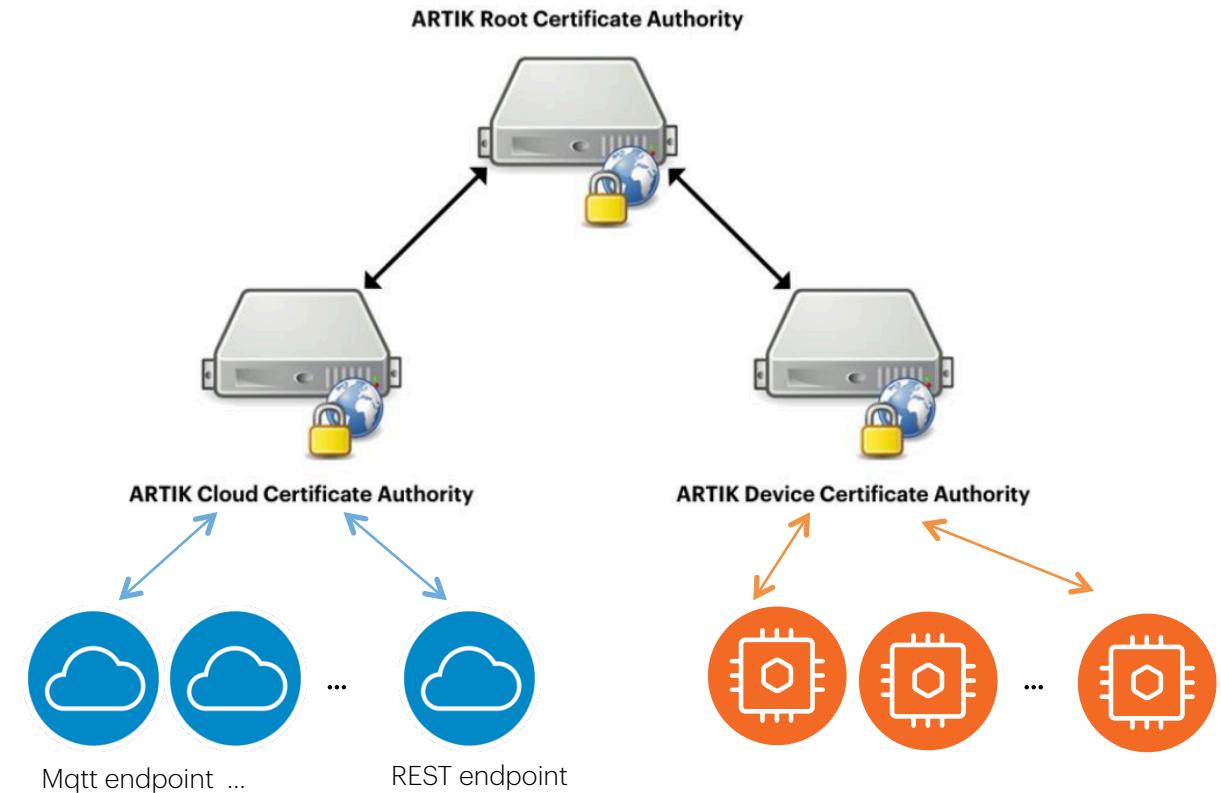
Different keys are used to encrypt and decrypt messages

Authentication



ARTIK PKI

- A Public Key Infrastructure (PKI) supports the distribution and identification of public encryption keys, establishing authenticity and trust in a system.
- ARTIK provides its own PKI, which is used to generate and apply unique certificates and key pairs to each ARTIK Module during manufacturing, and supports mutual authentication.
- PKI's core concept is (Digital) Certificate. Issued by a **Certificate Authority**, e.g, GlobalSign, Symantec
- ARTIK Root CA



Certificate

- A Certificate contains an identity (a hostname, or an organization, or an individual), a public key, signature etc.
- X.509 is a standard that defines the format of public key certificates.

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

01:00:17:03:07:00:00:00:04

Signature Algorithm: ecdsa-with-SHA256

Issuer: C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK High Security Device CA, CN=ARTIK High Security Device CA

Validity

Not Before: Mar 7 02:27:05 2017 GMT

Not After : Mar 7 02:27:05 2028 GMT

Subject: C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK High Security Device, CN=SIP-OP5WRS30 (01001703-0700-0000-041e-0e363c7eb564)

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:75:a5:0e:65:b8:31:40:66:e6:20:63:88:7c:dc:
78:d7:17:23:67:0e:79:4d:de:61:65:93:b0:50:a1:
19:1a:ce:1c:22:d3:ae:11:24:80:ee:96:d5:14:0f:
e0:bc:bc:a7:fa:8f:50:8e:35:2f:bc:db:ed:4b:1c:
fd:35:71:88:7e

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Key Usage: critical
Digital Signature, Non Repudiation

X509v3 Extended Key Usage:

TLS Web Client Authentication, TLS Web Server Authentication

Signature Algorithm: ecdsa-with-SHA256

30:45:02:21:00:ba:87:ec:ce:7e:83:d1:ec:6b:6
92:6f:f7:4a:d4:6d:19:4a:5d:e0:df:3d:0e:73:ef:
16:02:20:60:ee:16:f9:e5:e0:24:61:04:d6:25:09:5d:c7:87:
68:06:7c:e5:b3:ef:3e:4b:06:d1:5d:90:58:c0:b0:5f:ed

Issuer

Subject Information

Issuer Policies

Issuer Signature

Mutual Authentication

- Each ARTIK module is provisioned with:
 - An unique private key
 - Its associated certificate containing the public key
 - An ARTIK Root CA certificate
- ARTIK Cloud's server certificate is also rooted to the ARTIK Root CA certificate
- At connect time, ARTIK module and cloud exchange certificates for mutual authentication

Post Provisioning

- If you want to connect your ARTIK Module to a 3rd party Cloud service or implement a secure link between ARTIK modules, you can use your own certificate/key-pair
- The certificate/key-pair can be post provisioned into Secure Element(Secure Storage)

Transport Layer Security(TLS)

- TLS is a cryptographic protocol designed to provide communication security.
- TLS consists of two layers.
 - One layer allows client/server mutual authentication(uses X.509 certificates), and negotiates encryption algorithm and cryptographic keys.
 - The other layer provides connection security
- DTLS(Datagram Transport Layer Security) is an implementation of TLS over UDP
- OpenSSL, mbedTLS (TLS Implementation; cryptographic library; X.509 certificate handling library etc.)

Secure Communication

ARTIK Modules support:

- A unique key/certificate pair from ARTIK PKI
- True random number generator
- TLS/DTLS library for creating the channel
- Cryptographic Engine which takes advantage of the hardware acceleration for specific cryptographic operations

Samsung ARTIK™ 053/053s, 055s Wi-Fi® edge nodes

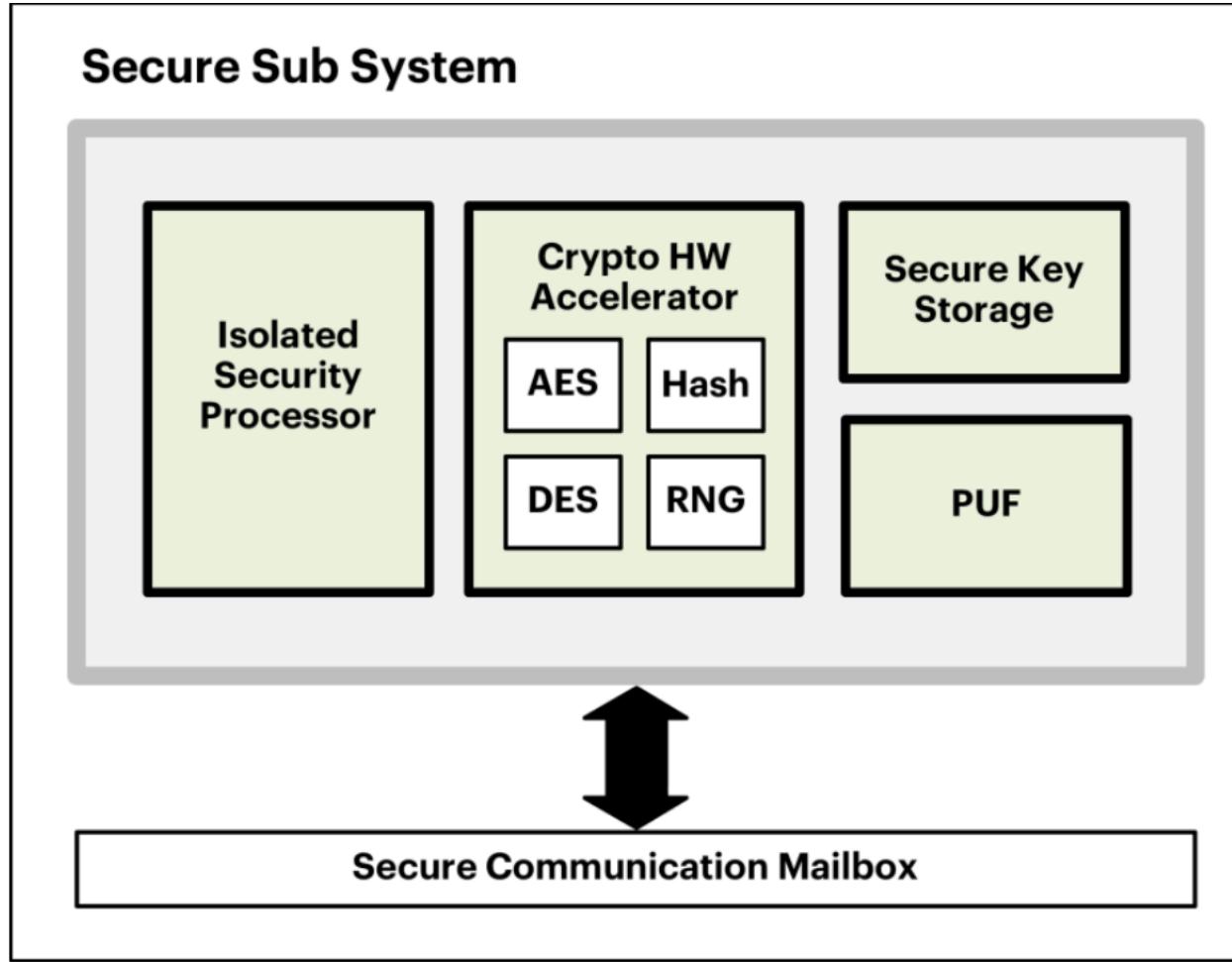


- Home health monitors, AEDs, fitness equipment, CPAP
- Smoke detectors, thermostats, energy monitors, appliances
- Sensors, lighting controllers, motors, valves
- Access control, fire monitors, smart switches

Processor	Main: ARM Cortex® R4 @ 320 MHz WLAN: ARM Cortex® R4 @ 480 MHz Security: ARM Cortex M0
Memory	RAM: 1.4 MB Flash: 8 MB SPI Flash on module
Connectivity	WLAN (Wi-Fi): IEEE 802.11 b/g/n
Regulatory	FCC (US), IC(Canada), CE(EU), KC(Korea), SRRC(China)
I/O	2xSPI, 5xUART (2-pin), 4xI2C, 7xPWM, 28xGPIO, 1xJTAG, 4xADC
Operating voltage	053, 053s: 5-12 VDC; 055s: 3.3 VDC
Temperature	-20° to 85° (°C)
Size	055s: 15 mm W x 26 mm H x 3.9 mm D 053, 053s: 15 mm W x 40 mm H x 3.9 mm D
Security	Secure Subsystem, Hardware-protected key storage with secure mutual authentication and data transfer, secure boot*, KMS* *S-versions only



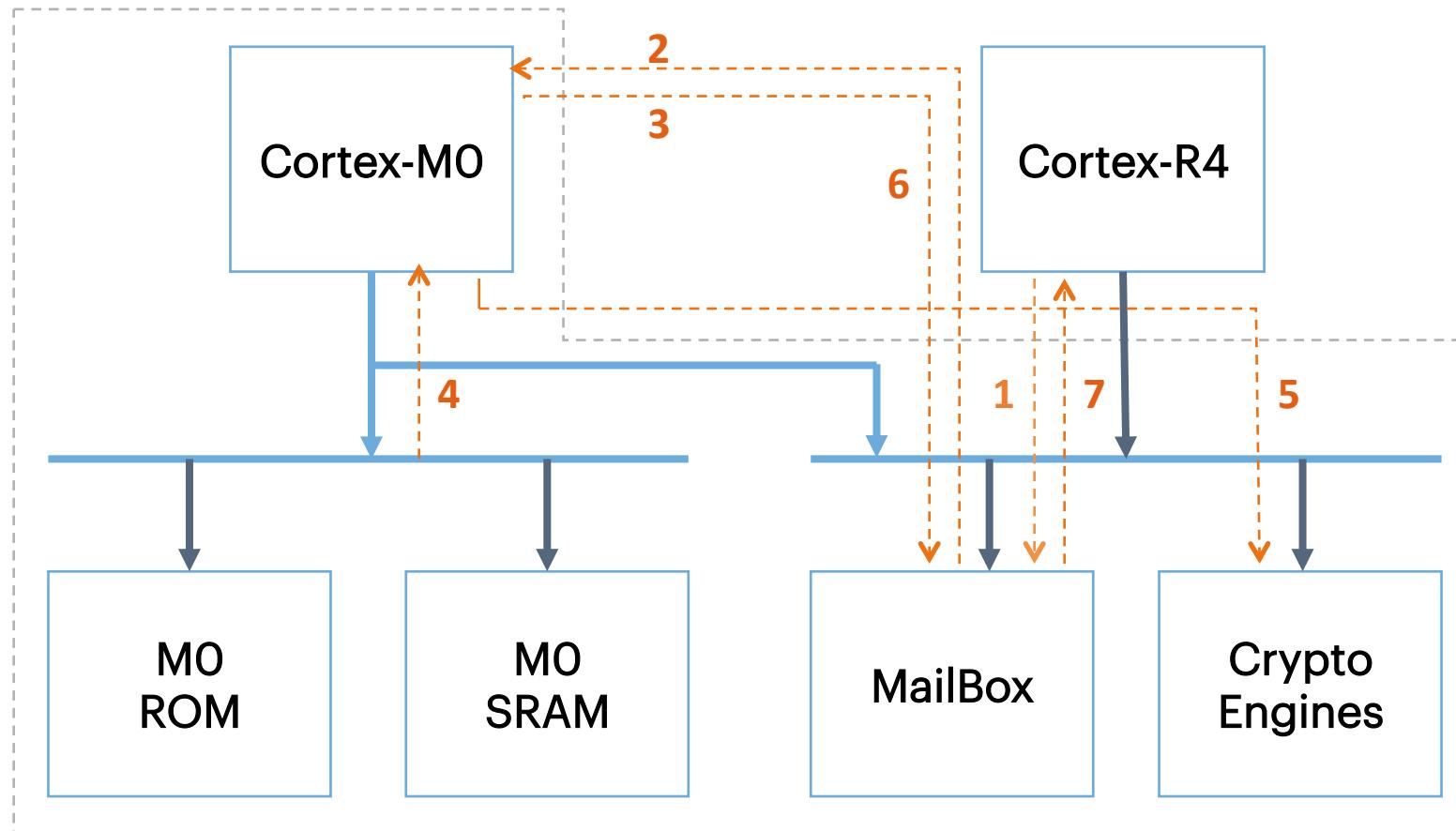
ARTIK 05x Security Subsystem



- Isolated Security Processor
- Cryptographic Hardware Acceleration
- A Physical Uncloneable Function(PUF)
- Secure Key Storage

Isolated Security Processor

Security Subsystem



Cryptographic Hardware Acceleration

Support for high performance cryptographic acceleration

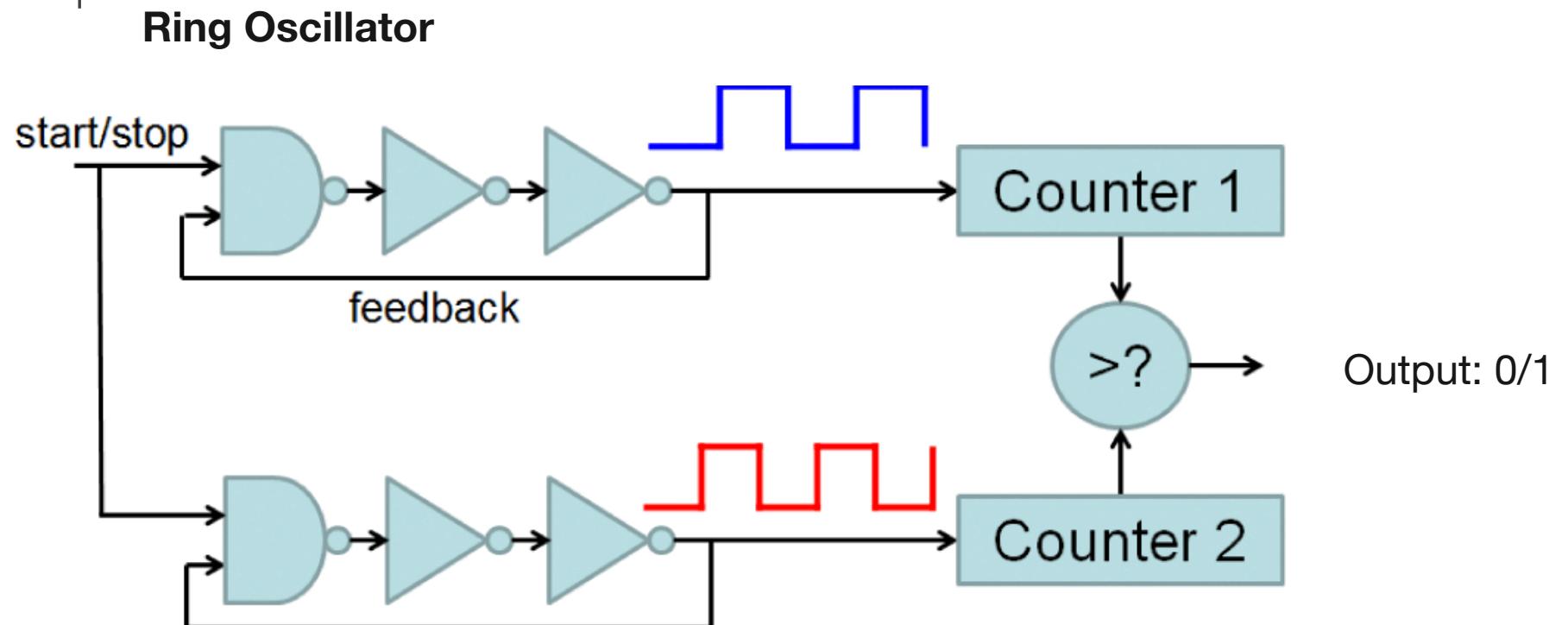
- Random Number Generation: DTRNG, PRNG
- Block Cipher: Secure AES, DES
- Hash Function: SHA1/SHA2/SHA3 with HMAC
- Public Key Cryptosystem: RSA, ECDSA, DH, ECDH
- FIPS Compliant: CAVP, CMVP, MDFPP

PUF (Physically Unclonable Function)

- Create a cryptographic key(PUF KEY) that can not be cloned by anybody else
 - PUF Key is auto generated using process variation during Manufacturing
 - Unchanging value over product lifetime
 - Unclonable
- Applications of PUF:
 - Key generation
 - Device identification
 - IP Protection
 - Protocols with challenge-response pairs

RO Frequency PUF

- RO (Ring-Oscillator) frequency is used as the PUF input to generate a unique key for each chip



Secure Communication

	ARTIK module (530, 710)	ARTIK S-module (530s/710s)	ARTIK module (053)	ARTIK S-Module (053s/055s)	Comments	
Secure communication	Per device unique key & cert	✓	✓	✓	✓	Uniquely identifies device
	Key stored in HW secure storage (secure element or secure key storage)	✓	✓	✓	✓	Secure key storage
	PKI infrastructure: Mutual authentication of device and cloud	✓	✓	✓	✓	Device communicates to authorized cloud and vice versa
	Post Provisioning		✓		✓	Provision with your own keys and certificates
Device protection/ secure code execution	KMS infrastructure for code signing		✓		✓	Key Management Service
	Code verification key in HW		✓		✓	Secure key storage
	Secure boot (verified boot on 530s/710s)		✓		✓	Boot image verification
	JTAG access locked		✓		✓	Lock out debug access
Data protection/ Secure storage	Secure OS (separate normal & secure operations)		✓			Hardware enforced secure applications via TEE
	Security Lib API (27 API calls)	get_cert(), gen_rand(), get_sig()	✓	get_cert(), gen_rand(), get_sig()	✓	Key Manager, Authentication, Secure Storage, Post Provisioning, Encrypt/Decrypt
	Secure storage		✓		✓	Secure Element & Encrypt data stored on Flash

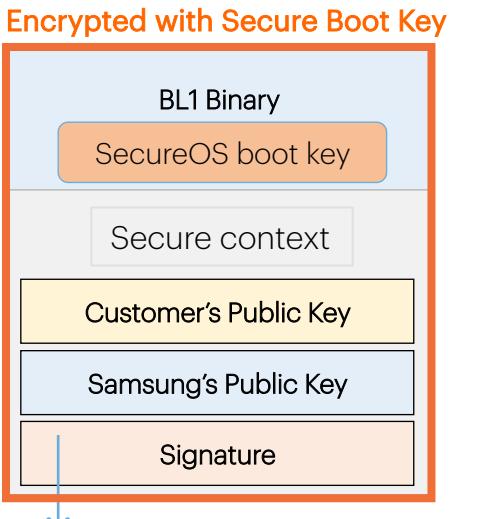
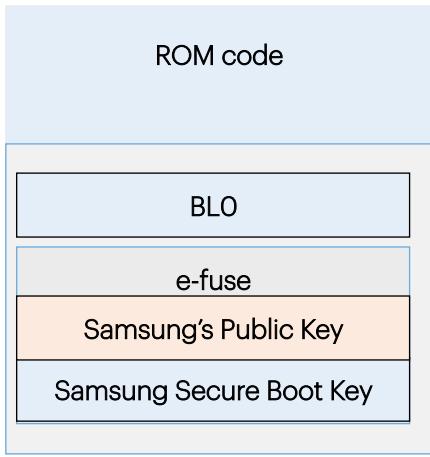


Secure storage

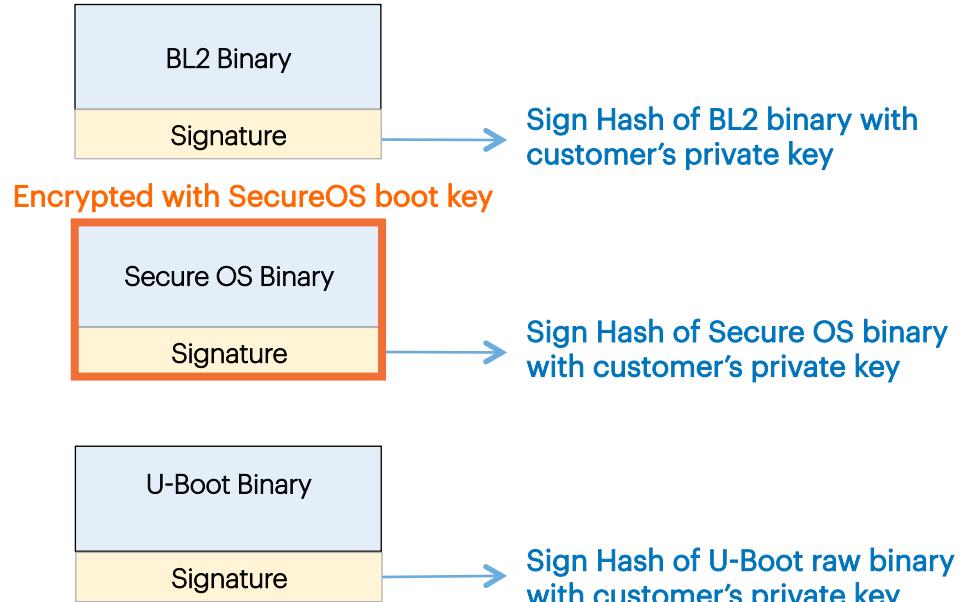
Secure Boot

- Secure Boot ensures only authenticated software runs on the device
- Authentication is based on digital signature verification.
- To achieve secure boot, processor/SoC support is required.

Secure Boot for ARTIK



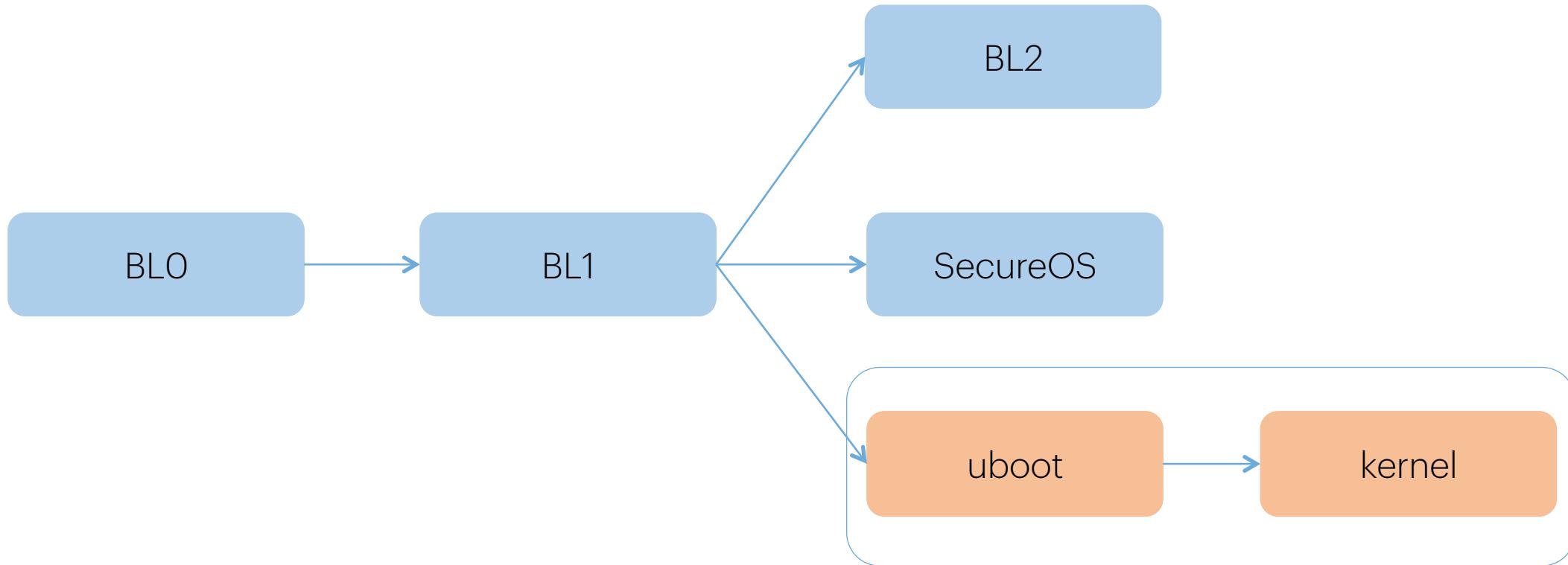
- Decrypt BL1
- Verify Samsung's Public Key with e-Fuse
- Verify signature with hash of BL1 raw binary and customer's public key with confirmed Samsung's Public Key



ROM Code – Root of Trust

- Needs to store the public keys which will be used to decrypt the signature of the 1st stage bootload. The public keys are stored in non-volatile memory.
- One-Time-Programmable(OTP) fuses are used. It is irreversible
- OTP fuses are silicon-expensive in terms of space and store a relatively small amount of information
- A public key is at least 1KB. It is less expensive to store only the hash of the public key in OTP, then compare it to the hash of the public key embedded in a given binary

Secure Boot for ARTIK – cont.



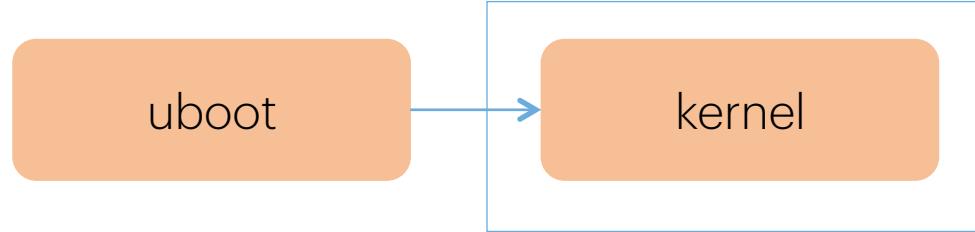
Provided as image, customer can not modify

Provide source code, customer can modify

Chain of Trust

- Extending the trust scheme to user space involves establishing a chain of trust.
- Verified Boot establishes a full chain of trust, starting from a hardware-protected root of trust. Each element in the boot process verifies the integrity and authenticity of the next stage before handing over execution.
 - ROM verifies signed bootloader, bootloader verifies signed kernel and kernel verifies/ mounts signed root filesystem (RFS).

FIT Image

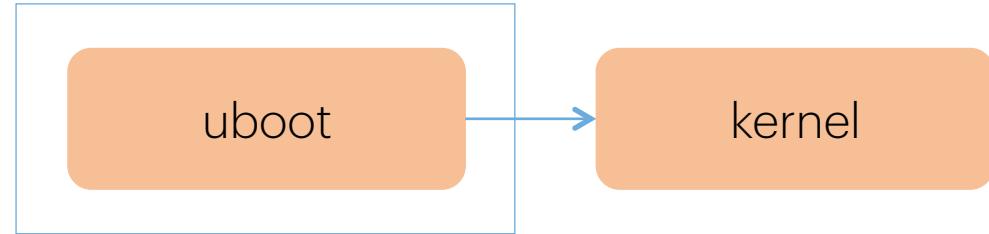


- FIT stands for Flattened Image Tree. It is a single binary that contains all the images required to boot up a system: kernel, device tree and ramdisk etc.
- FIT Image is then signed using a private key, and the signature is embedded inside the FIT image.
- Public key is embedded as part of U-boot device tree.
- Since U-boot is authenticated, we can trust the public key inside the FIT Image.

FIT Image Tree Source File(ITS)

```
/dts-v1/;
{
    description = "ARTIK530s Image with kernel and FDT blob";
    #address-cells = <1>;
    images {
        kernel@1 {
            description = "Linux kernel";
            data = /incbin/("./zImage");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "none";
            load = <0x91080000>;
            entry = <0x91080000>;
            signature@1 {
                algo = "sha1,rsa2048";
                key-name-hint = "dev";
            };
        };
        fdt@1 {
            description = "Flattened Device Tree blob";
            data = /incbin/("./s5p4418-artik530...dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            load = <0x9b000000>;
            signature@1 {
                algo = "sha1,rsa2048";
                key-name-hint = "dev";
            };
        };
    };
    ramdisk@1 {
        description = "RAMDISK";
        data = /incbin/("./uInitrd.fit");
        type = "ramdisk";
        arch = "arm";
        os = "linux";
        compression = "none";
        load = <0x9a000000>;
        entry = <0x9a000000>;
        signature@1 {
            algo = "sha1,rsa2048";
            key-name-hint = "dev";
        };
    };
    configurations {
        default = "conf@1";
        conf@1 {
            description = "Boot kernel with FDT blob";
            kernel = "kernel@1";
            fdt = "fdt@1";
            ramdisk = "ramdisk@1";
        };
    };
}
```

U-Boot Device Tree Blob



- U-Boot has Device Tree Blob(DTB) support, used the same way the kernel does to probe drivers
- DTB can be used to store a public key for kernel image signature verification
- DTB is appended to the U-boot binary

```
$ vi ${HOME}/artik530s/u-boot-artik/arch/arm/dts/s5p4418-artik530-raptor.dts
/
+     signature {
+         key-dev {
+             required = "conf";
+             algo = "sha1,rsa2048";
+             key-name-hint = "dev";
+         };
+     };
}
```

Root filesystem and application data protection

- If a root filesystem is read-only and small enough to run out of RAM, we can embed the root filesystem inside the FIT Image for authentication.
- For typical rootfs, we need to encrypt/verify the contents of the entire disk at a block level. This is performed by the kernel's device mapper(dm) modules.
- Device-Mapper is the infrastructure in the Linux kernel to create virtual layers of block devices.
- Device-Mapper verity(dm-verity): provides integrity checking of block devices using kernel crypto API

Code Signer (Development Stage)

```
Invoking: ARTIK GCC Create Head Bin
C:/ARTIK/SDK/A055s/v1.7.1/common/tools/s5jchksum.py      "tinyara.bin"
"tinyara_head.bin"

Finished building: tinyara_head.bin

Invoking: ARTIK GCC Create Head Sign
C:/ARTIK/SDK/A055s/v1.7.1/common/codesigner/artik05x_AppCodesigner
C:/ARTIK/SDK/A055s/v1.7.1/common/codesigner/rsa_private.key
"tinyara_head.bin"

. Seeding the random number generator...
. Reading private key from
'C:/ARTIK/SDK/A055s/v1.7.1/common/codesigner/rsa_private.key'
. Generating the RSA/SHA-256 signature
. Done (created "tinyara_head.bin-signed")

+ Press Enter to exit this program.

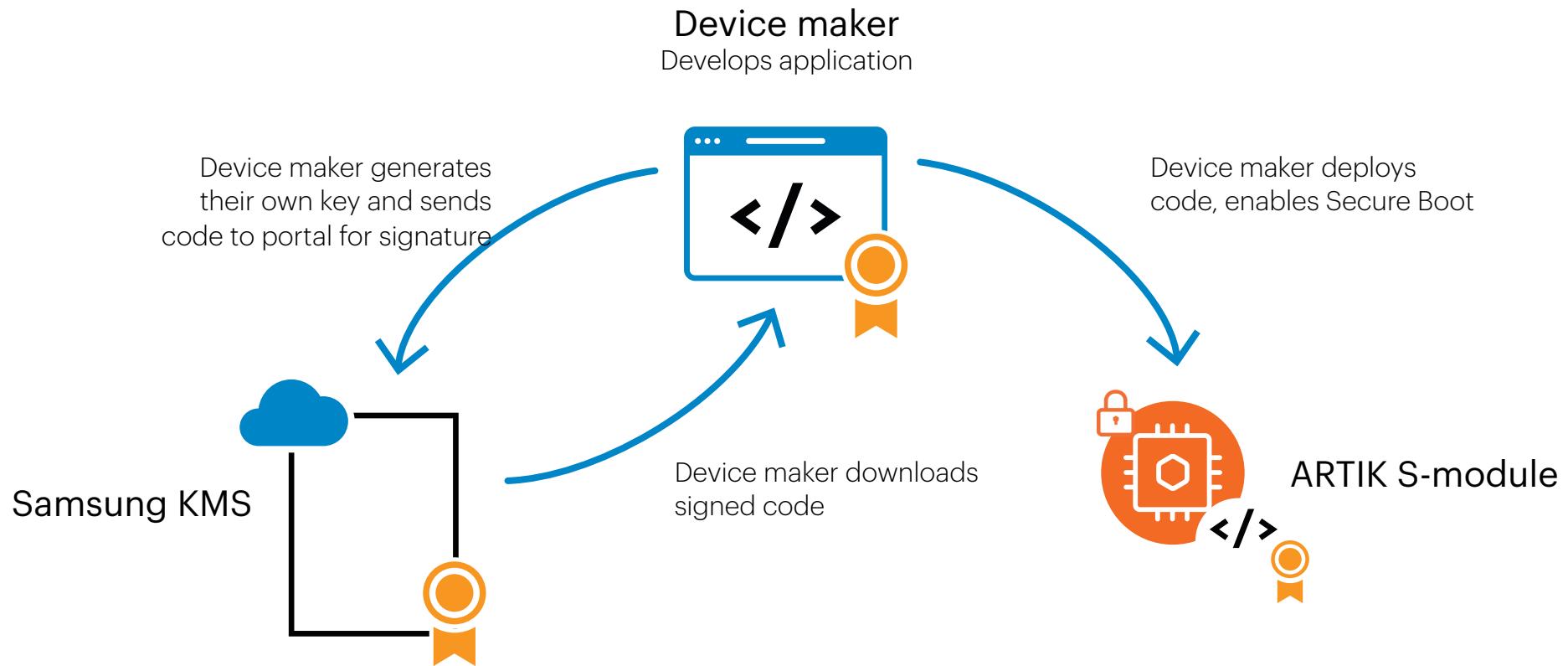
Finished building: tinyara_head.bin-signed
```



NEW

Samsung ARTIK™ Key Management System(KMS)

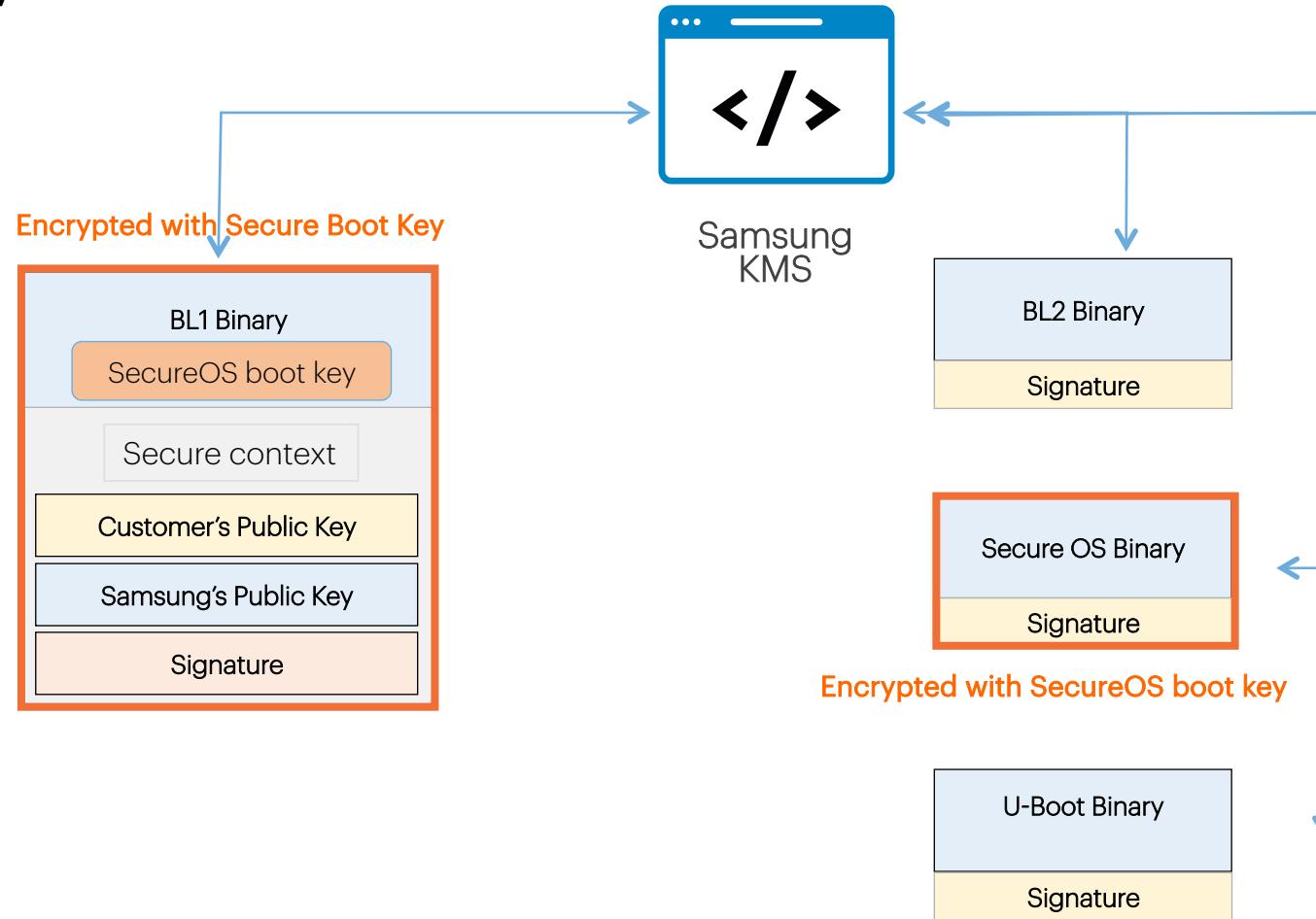
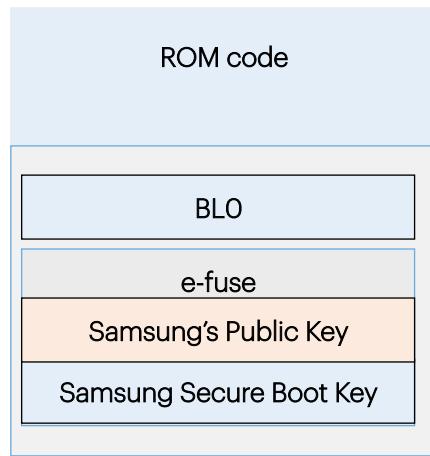
Code signing portal manages signing key



Key Management System

- Signing keys are stored and operated within FIPS 140-2 certified Hardware security modules (HSM)
- Images are signed through a highly secure cryptography standard (SHA-256 w/ RSA2048 encryption)
- Strict access control policies
- Only accessible through whitelisted IP addresses

Key Management System Role



KMS Key Management

The image displays three screenshots of the KMS Key Management interface:

- Create a new key (Stage 1):** A dropdown menu shows device models: ARTIK_520s, ARTIK_530s_530s-1G, ARTIK_710s, and ARTIK_053s_055s. The last option is selected. The "New Key Name" field contains "ARTIK_053s_055s". The "Description" field is empty. A success message at the bottom says "Success! Key \"ARTIK_053s_055s\" was created successfully."
- Create a new key (Stage 2):** The "Model" field is set to "ARTIK_053s_055s". The "New Key Name" field contains "055s-key01". The "Soft Card Password" field contains a masked password. The "Description" field contains "Key for Partner Workshop".
- Key Management:** A table lists three keys:

Model	Key Name	Public Key	Creation Time	Description
ARTIK_520s	artikaura01-5...	artikaura01-520test.spk	2017/07/24 14:04:42	
ARTIK_710s	artikaura01-7...	artikaura01-710test.spk	2017/07/24 14:09:05	
ARTIK_053...	artikaura01-0...	artikaura01-055s-key01.spk	2018/02/02 09:26:15	Key for P...

Stage 1:

- Public key is immediately available on KMS portal
- Send ARTIK team the resulting public key.
- ARTIK team signs the bootloader stage 1 (BL1) image and deliver it to you by e-mail.



KMS File Management (Stage 2 Images)

Upload bootloader stage 2(BL2) and OS files for self-signing

Upload

Model: * ARTIK_053s_055s

File name: tinyara_head.bin

Description: Tinyara head bin for ARTIK 055s
.....
31 / 255 characters written

UPLOAD CANCEL

File Management

Success! File "tinyara_head.bin" uploaded.

UPLOAD EDIT DELETE

	Model	Source ...	Signed File	Sign Key Name	Upload Time	Sign Time	Description
<input type="checkbox"/>	ARTIK_053...	tinyara_head.b	tinyara_head.bin-sig	artikaura01-055s-...	2018/02/02 10:02:48	2018/02/02 10:03:05	Tinyara fo
<input type="checkbox"/>	ARTIK_530...	logo.png	No Key Available	-	2017/07/24 14:04:16		
<input type="checkbox"/>	ARTIK_053...	tinyara_head.b	SIGN	-	2018/02/05 03:46:30		Tinyara h



KMS File Management (Stage 2 Images)

Sign BL2/OS image with generated key, and download the signed BL2/OS images.

File Management

Sign

Model: ARTIK_053s_055s

Source File: tinyara_head.bin

Sign Key Name: * artikaura01-055s-key01

Soft Card Password: *

SIGN CANCEL

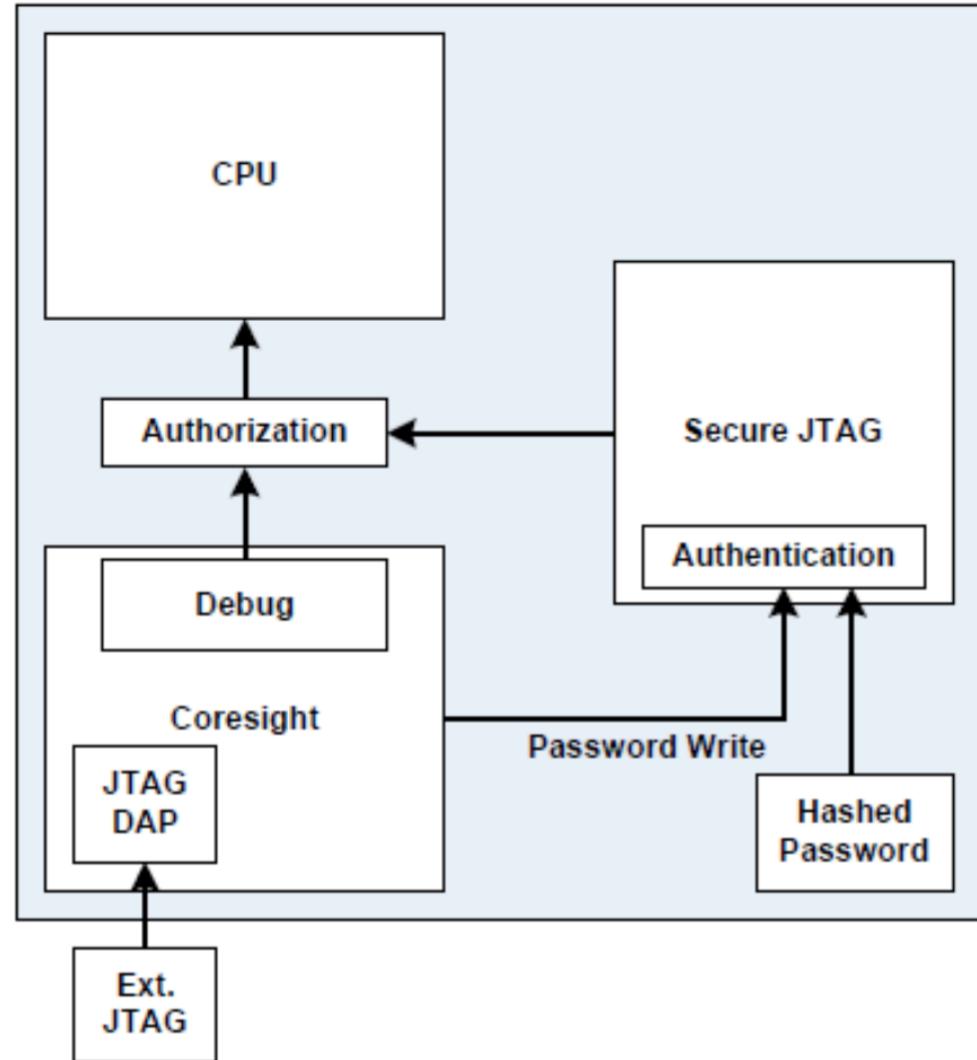
File Management

Success! File "tinyara_head.bin" signed.

UPLOAD	EDIT	DELETE					
	Model	Source ...	Signed File	Sign Key Name	Upload Time	Sign Time	
<input type="checkbox"/>	ARTIK_053s_055s	tinyara_head.b	tinyara_head.bin-signed	artikaura01-055s-...	2018/02/02 10:02:48	2018/02/02 10:03:05	
<input type="checkbox"/>	ARTIK_530s_530...	logo.png	No Key Available	-	2017/07/24 14:04:16		
<input type="checkbox"/>	ARTIK_053s_055s	tinyara_head.b	tinyara_head.bin-signed	artikaura01-055s-...	2018/02/05 03:46:30	2018/02/05 03:49:31	

Secure JTAG

- Secure JTAG is disabled by default.
- Secure JTAG can be enabled to an OEM, where it requires a password to authenticates and authorizes JTAG access. The password is based on the serial number of module.
- The information is only made available through an authorized request to Samsung.



Device Protection

	ARTIK module (530, 710)	ARTIK S-module (530s/710s)	ARTIK module (053)	ARTIK S-Module (053s/055s)	Comments
Secure communication	Per device unique key & cert	✓	✓	✓	Uniquely identifies device
	Key stored in HW secure storage (secure element or secure key storage)	✓	✓	✓	Secure key storage
	PKI infrastructure: Mutual authentication of device and cloud	✓	✓	✓	Device communicates to authorized cloud and vice versa
	Post Provisioning		✓	✓	Provision with your own keys and certificates
Device protection/ secure code execution	KMS infrastructure for code signing		✓	✓	Key Management Service
	Code verification key in HW		✓	✓	Secure key storage
	Secure boot (verified boot on 530s/710s)		✓	✓	Boot image verification
	JTAG access locked		✓	✓	Lock out debug access
Data protection/ Secure storage	Secure OS (separate normal & secure operations)		✓		Hardware enforced secure applications via TEE
	Security Lib API (27 API calls)	get_cert(), gen_rand(), get_sig()	✓	get_cert(), gen_rand(), get_sig()	Key Manager, Authentication, Secure Storage, Post Provisioning, Encrypt/Decrypt
	Secure storage		✓	✓	Secure Element & Encrypt data stored on Flash



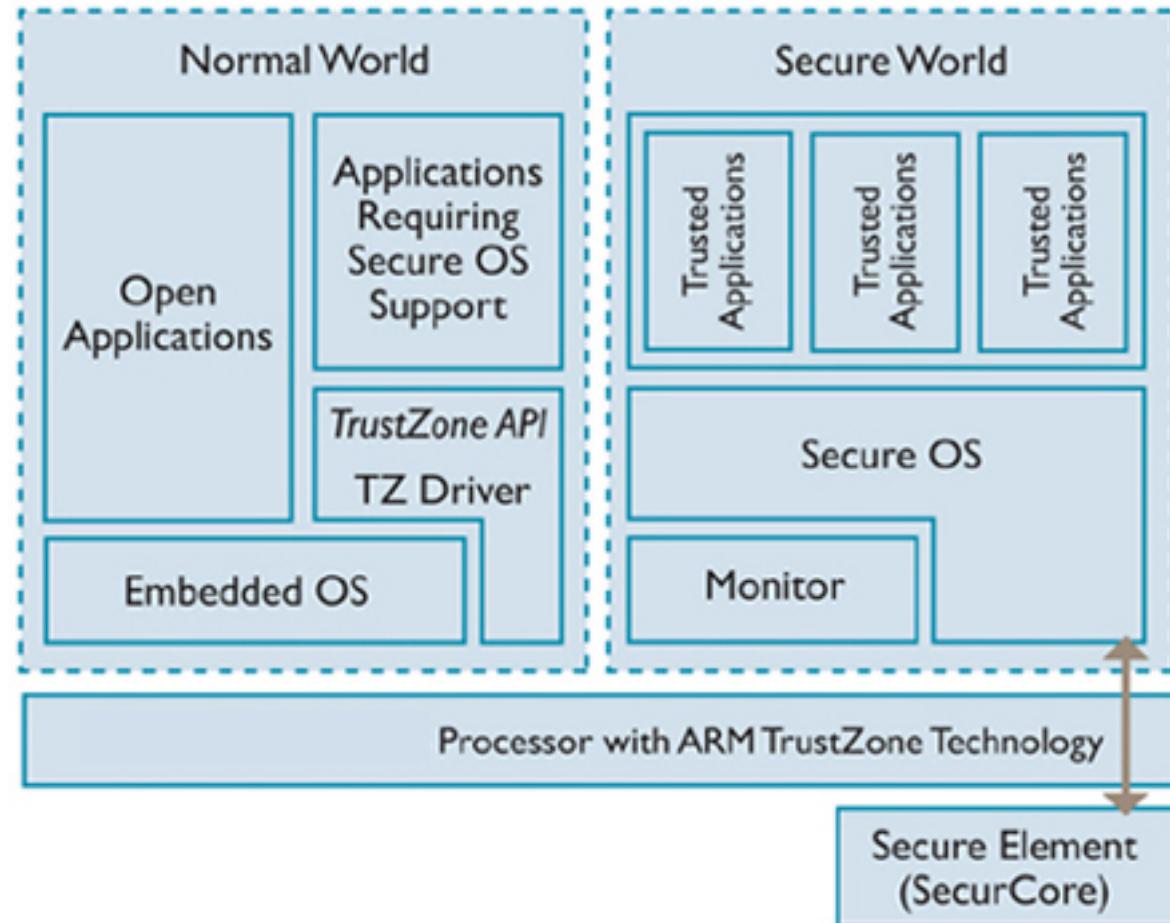
Secure storage

ARM TrustZone

- ARM TrustZone technology provides system-wide isolation for trusted software
- It enables two separate environments: Rich Execution Environment and Trusted Execution Environment. Partition the SoC's hardware and software resources so they exist in one of the environments
- Both execution environments have the same capabilities, but operate in a separate memory space
- A single physical processor can execute from both the Normal world and the Secure world in a time-sliced fashion.
- TrustZone is integrated on Cortex-A, Cortex-M23, Cortex-M33 processors

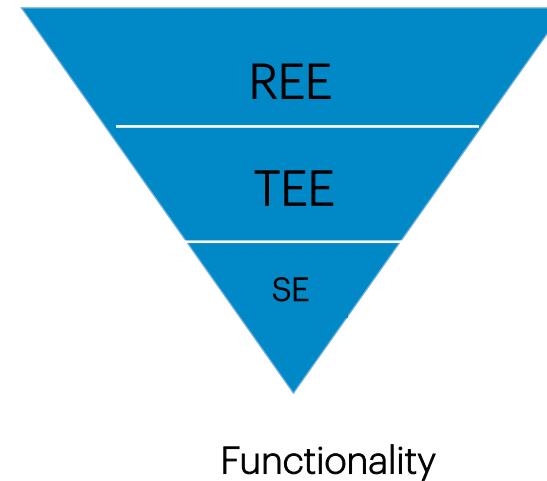
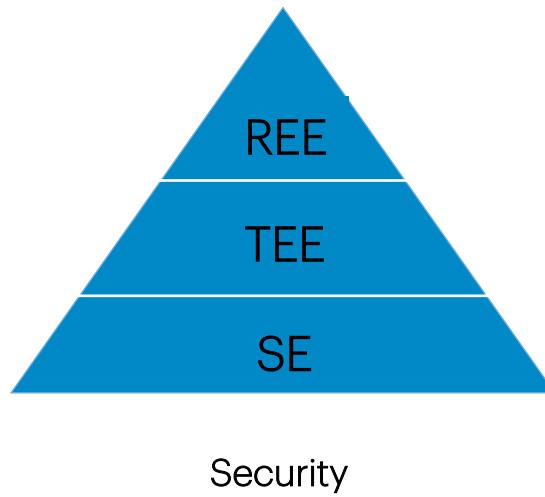
Trusted Execution Environment on ARTIK 5/7x (TEE)

- ARTIK 530s and 710s modules support Trusted Execution Environment(TEE), based on Arm TrustZone
- TEE guarantees that the code and data within are protected with regards to confidentiality and integrity
- Trusted Applications running in a TEE have access to the full power of a device's main processor and memory, while hardware isolation protects them from user applications running in rich OS.



TEE vs. SE

- TEE offers a higher level of performance and functionality than a Secure Element(SE), TEE uses a hybrid approach that uses both hardware and software to protect data.

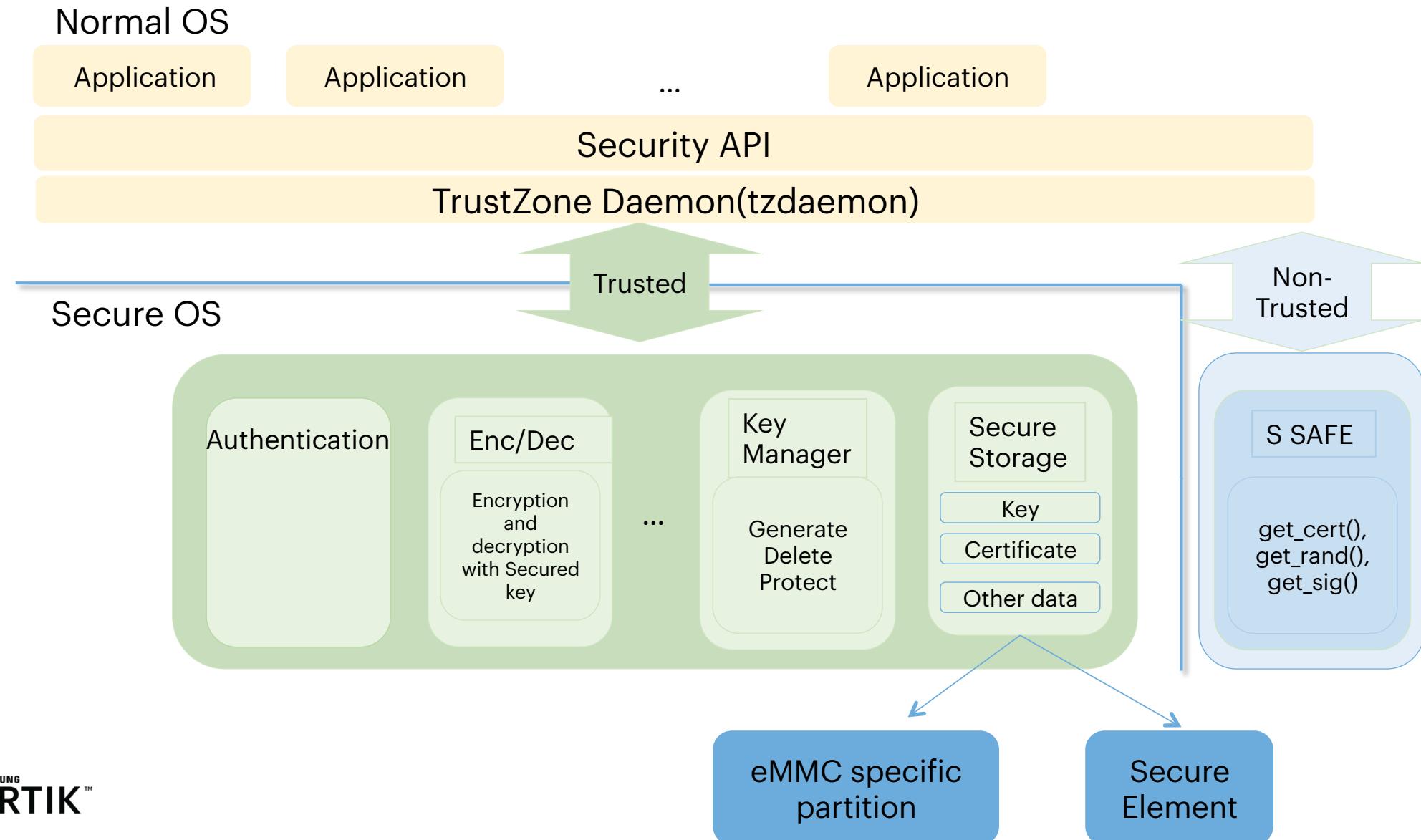


Trustonics TEE and SDK

- GlobalPlatform is an organization that provides software APIs, compliance and certification schemes for the TEE.
- TEE providers: Trustonics, Op-TEE etc.
- ARTIK supports Trustonics TEE
- Trustonics injects a Root of Trust into devices TEE during manufacturing stage, which is used for TA authentication
- Trustonics Kinibi
- Trustonics SDK



ARTIK Security Architecture



ARTIK Security APIs

Category	ARTIK API	Description
Initialize	see_init see_deinit	For Initializing a new SEE session.
Key Management	see_generate_key see_set_key see_get_pubkey see_remove_key	For management of symmetric and asymmetric keys that are not exposed to non-secure operating system user space.
Authentication	see_generate_random see_generate_certificate see_set_certificate see_get_certificate see_get_rsa_signature see_verify_rsa_signature see_get_ecdsa_signature see_verify_ecdsa_signature see_get_hash, see_get_hmac see_generate_dhparams(ecdhkey)	For use with the TLS library with callback functions to generate a digital true random number, generate or get certificates, get signatures and verify, and to generate or compute DHM parameters.

ARTIK Security APIs

Category	ARTIK API	Description
Secure Storage	see_read_secure_storage	For access to the secure storage for data, credentials, and keys.
	see_write_secure_storage	
	see_delete_secure_storage	
	see_get_size_secure_storage	
	see_get_list_secure_storage	
Post Provision	see_post_provision	For injecting an HMAC key or asymmetric key pair (ECC/RSA) into the secure element.
	see_post_provision_lock	
Encryption/ Decryption	see_aes_encryption	For Data encryption and decryption using a key in secure storage.
	see_aes_decryption	
	see_rsa_encryption	
	see_rsa_decryption	

Secure Storage – Secure Element

Secure Element – an isolated storage device that supports 2 slots of ECDSA key pairs (16 AES 128-bit keys).

- The Secure Element provides high levels of hardware security with anti-tamper measures.
- It includes cryptographic services such as random-number generation, key/data secure storage, and certificates handling and processing.
- All communication from the Secure Element to the processor is secured and encrypted.
- Uses Smart Shield, Smart Sensor, Smart Core etc. technologies to achieve the highest level of security and protection.
- The Secure Element meets the Common Criteria (CC) certification for security and for Evaluation Assurance Level (EAL) 5.

Secure Storage – eMMC file system

- eMMC file system (Flash-based)
 - Uses the same storage as the normal operating system. However, a specific partition is managed by Secure OS.
 - All data in this partition is encrypted with a unique key generated at run time, and is stored as a file unit of 32KB with a maximum of 1024 files that may be stored.
 - Applies to ARTIK 05x and 5/7 modules.

Data Protection

	ARTIK module (530, 710)	ARTIK S-module (530s/710s)	ARTIK module (053)	ARTIK S-Module (053s/055s)	Comments
Secure communication	Per device unique key & cert	✓	✓	✓	Uniquely identifies device
	Key stored in HW secure storage (secure element or secure key storage)	✓	✓	✓	Secure key storage
	PKI infrastructure: Mutual authentication of device and cloud	✓	✓	✓	Device communicates to authorized cloud and vice versa
	Post Provisioning		✓	✓	Provision with your own keys and certificates
Device protection/ secure code execution	KMS infrastructure for code signing		✓	✓	Key Management Service
	Code verification key in HW		✓	✓	Secure key storage
	Secure boot (verified boot on 530s/710s)		✓	✓	Boot image verification
	JTAG access locked		✓	✓	Lock out debug access
Data protection/ Secure storage	Secure OS (separate normal & secure operations)		✓		Hardware enforced secure applications via TEE
	Security Lib API (27 API calls)	get_cert(), gen_rand(), get_sig()	✓	get_cert(), gen_rand(), get_sig()	Key Manager, Authentication, Secure Storage, Post Provisioning, Encrypt/Decrypt
	Secure storage		✓	✓	Secure Element & Encrypt data stored on Flash

Security Questionnaire

How do you provide security across all attack surfaces?

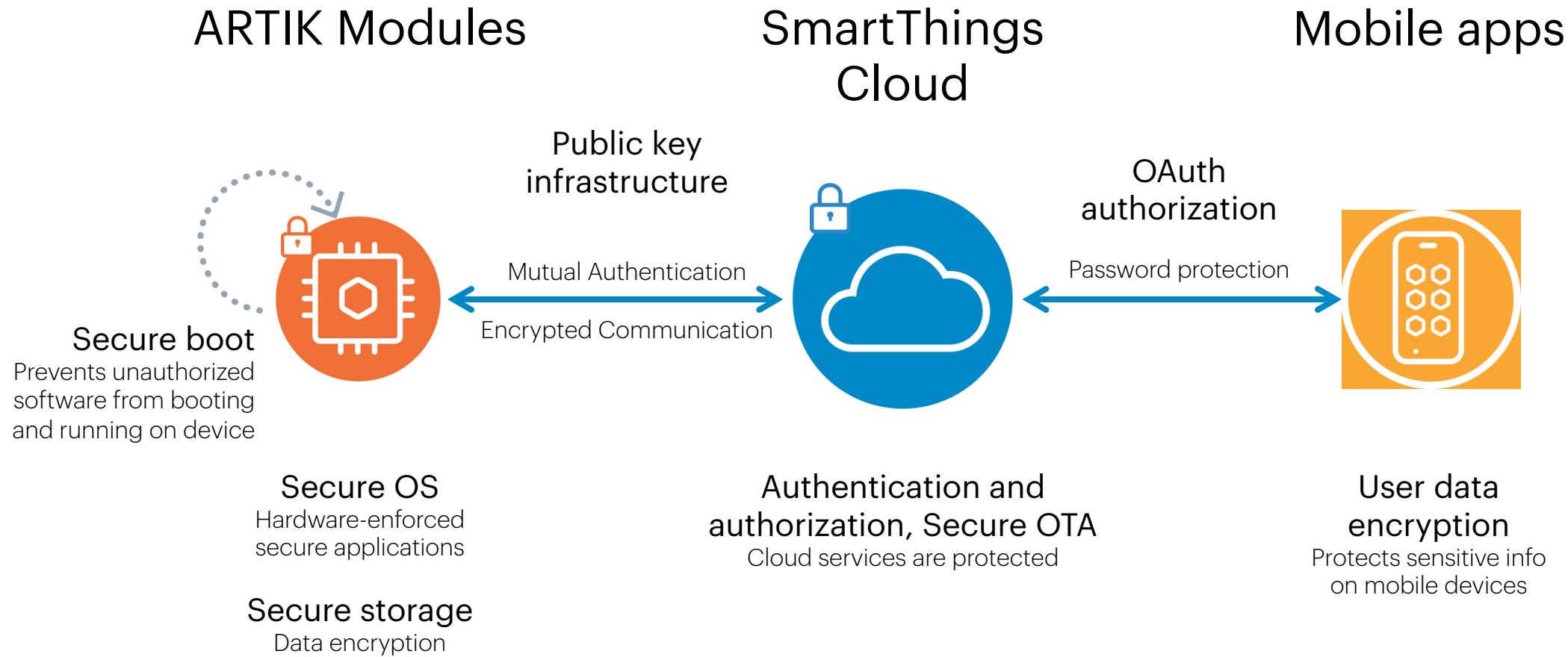
Question	ARTIK 5/7/053
Do you support secure communication from device to device or device to cloud? How do you secure communication? Are you using TLS1.2 or higher?	Yes, HTTPs using TLS 1.2
How do you establish identity of device?	Using unique certificate on each device
How does the device establish identity of cloud?	Both device and cloud are chained to ARTIK Root CA and can verify each other certificates
Do you have mutual authentication when enabling secure communication?	Yes
Do you have the infrastructure to inject unique key and certificate in each device to establish unique identity per device? How much does it cost?	Yes (Done at Samsung factory. Cost included in module)
How do you protect your certificate, keys?	Specialized HW on module (secure element or encrypted eMMC partition)
Is your certificate infrastructure secure? How do you secure your Root Certificates? How much does it cost?	Yes (Root CA secured by 3 rd party security vendor)
How do you guarantee your firmware integrity?	Secure Boot

ARTIK Platform Security



Samsung ARTIK™ End-to-end Platform Security

End-to-end protection for you and your customers



Secure OTA

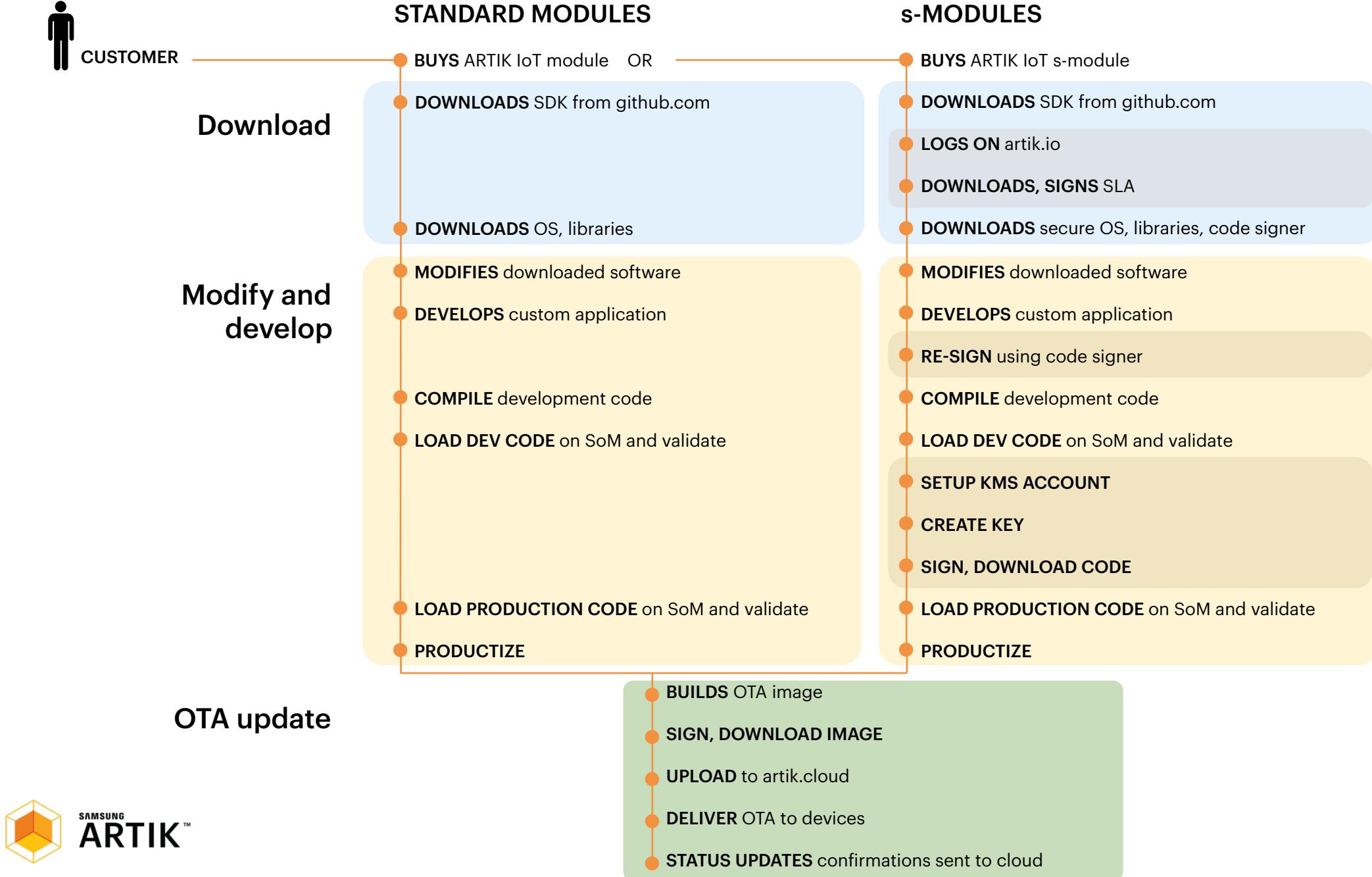
- ARTIK OTA solution uses LWM2M protocol over a TLS-secured link
- Resistance against redirect attacks
- Protection against DDOS attacks by ensuring only registered and authorized devices will be able to download new images
- Generation of a unique single-use URL for each device for a given update; a different URL is provided for each OTA update.

Samsung ARTIK™ End-to-end Platform Security*

	Feature	ARTIK
Modules	Secure element key storage, secure boot	✓ Included
	Security infrastructure: PKI and KMS	✓ Included
	Unique device ID and certificate	✓ Included
	Secure data storage with data encryption	✓ Included
Platform software	Secure device registration	✓ Included
	Secure OTA updates	✓ Included
Cloud Infrastructure	Supports HIPAA compliant solutions	✓ Included
	OWASP top 10	✓ Included
	Internal and external security audits	✓ Included
Cloud services	AAA (Authentication, Authorization, Accounting)	✓ Included
	API Security	✓ Included
	3 rd party device discovery and mutual authentication	✓ Included
	Data privacy management, identity, permissions,	✓ Included
Communications	TLS, VPN	✓ Included
	DTLS Application level security; BLE session security	✓ Included
Applications	Key and secure app data encryption and storage	✓ Included
	2-factor authentication; OAuth2; client side certificates	✓ Included

* Feature list is not exhaustive

S-Module Design Considerations



S-Module Considerations

- Can I use my own PKI?
- How to guarantee secure communication if I am using a 3rd party cloud?
- What's the cost of Secure Boot?
- Can I use my own KMS? What are the benefits of using Samsung KMS?
- To access secure OS on 5x/7x, should I use Trustonics SDK or ARTIK SDK?
- How can I leverage ARTIK Security APIs?
- How about Secure Device Registration and Secure OTA?

Use your own PKI

Customer can choose to use their own PKI

- **Secure Storage:** Customer can use Security APIs to set their own keys and certificates inside a trusted storage area where the keys are used for TLS services but can never be read out.
- **Secure Element:** For customers with volume, a second key and certificate set can be provisioned in the Secure Element itself, providing the "bullet-proof" non-volatile security to use the cloud and PKI of your choice.

Cost of Secure Boot

- Logistics and overall project complexity: whole architecture to create keys, sign the images with the keys..
- Signing private key protection
- Workflow complexity for developers: if the platform is locked down, need to re-sign the binary every time and validate the chain-of-trust
- Boot time (a couple of authentications to be made along the way until system boots up)

Choice of KMS

- How are signing keys saved in Samsung KMS?

Signing keys are saved in FIPS 140-2 certified Hardware security modules

- What is the signing algorithm used by Samsung KMS?

Signing Algorithm is SHA256wRSA2048

- How robust is Samsung KMS?
 - Only whitelisted IP can access Samsung KMS portal
 - Samsung KMS uses Thales backend.

Choice of KMS – cont.

- What if I choose to use my own KMS?
 - Customer needs to generate and manage their own signing keys.
 - Submit the public signing key to Samsung to sign BL1 image.
 - Samsung will share the code signing tool to customers under NDA for them to sign their BL2, bootloader, OS .. code.

Choice of Security APIs

- ARTIK Security APIs are offered as part of ARTIK SDK.
- Trustonics has an SDK for development of trusted applications. It requires cost/licensing and key injection in the process
- If customer needs to create Trusted Application(TA) running inside TEE, they need to use Trustonics TEE and TA. ARTIK SDK doesn't support TA at this point

Secure OTA

- Secure communication channels
- Code provenance and integrity checks are essential: Cryptographic code signing must be used to confirm that connected devices from verified sources, and the code has not been altered in transit
- Dual partition upgrade system allows critical boot images to be updated while still keeping a known-good copy of the image as backup

Memory, Boot time benchmark data

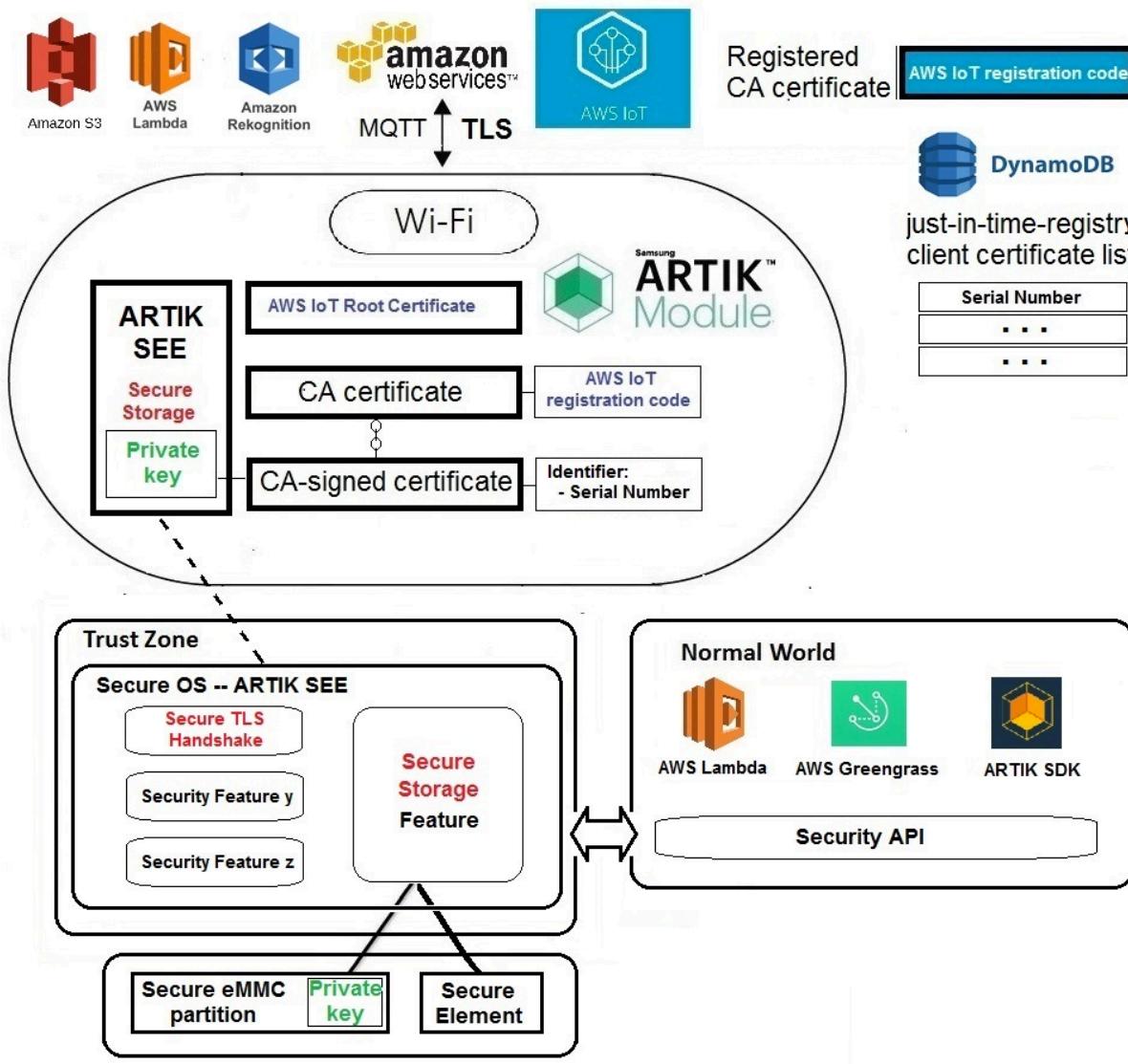
- A530s and A710s consumes ~ 100MB more eMMC storage compared to the standard versions.
- A530s and A710s consumes 1.5~2.5MB more DRAM compared to the standard versions
- Boot time difference between standard and secure modules are between 0.5~3.7 secs.

Documentation

- Security Topics Index Page: <https://developer.artik.io/documentation/security-topics/>
- Secure Boot: <https://developer.artik.io/documentation/advanced-concepts/secure-os/secure-boot.html>
- Secure OS: <https://developer.artik.io/documentation/advanced-concepts/secure-os/secure-os.html>
- Production KMS: <https://developer.artik.io/documentation/advanced-concepts/secure-os/kms.html>
- Mutual Authentication: <https://developer.artik.io/documentation/advanced-concepts/secure-os/secure-os.html>
- Security APIs: <https://developer.artik.io/documentation/security-api/>

Security Use Case & Demo

Connect to AWS



- Provisioning: The ARTIK device must be provisioned with a key, a CA certificate that has been registered to AWS IoT, and a client certificate (with serial number) that has been derived from the registered one.
- Just-in-time registration (JITR): The AWS IoT cloud must have a Lambda function that recognizes the connecting device, registers it
- During the mutually-authenticated TLS handshake, AWS IoT receives the device-unique certificate, extracts the serial number, and looks it up in the whitelist.

- Generate an RSA private key.

```
openssl genrsa -out rsaDevCert.key 2048
```

- Convert it to DER format and put it in Secure Storage.

```
openssl rsa -in rsaDevCert.key -outform DER -out rsaDevCert.der  
see_test -K set-key -a 4097 -k rsaDevCert.key -I rsaDevCert.der
```

- Use the key in Secure Storage to create a certificate signing request (CSR). Replace the subject information as you choose.

```
openssl req -new -engine artiksee -key "rsa2048://rsaDevCert.key" -keyform e -out  
"filename.csr" -subj "/O=company/OU=team/CN=marks-iot/C=US"
```

- Submit the CSR to your CA for signing, to create a device-unique certificate.
- openssl x509 -req -in filename.csr -CA rsaCACertificate.pem -CAkey
rsaCACertificate.key -CAcreateserial -out rsaDevCert.crt -days 365 -sha256 -
set_serial 0x123456