SECURE BY DESIGN SERIES

OP-TEE: Trusted Execution Environments on i.MX Processors

Nathan Barrett-Morrison
Senior Embedded Systems Engineer
Timesys Corporation

SECURE CONNECTIONS FOR A SMARTER WORLD

EXTERNAL

Agenda

TEE Overview

OP-TEE Overview

- Memory Protections
- Trusted Applications
- Enabling & Testing
- Enhanced OpenSSL Engine

Other Security Considerations

Trusted Execution Environment

(TEE) Overview

SECURE CONNECTIONS
FOR A SMARTER WORLD

Trusted Execution Environment (TEE) Overview

Why is TEE needed?

- Another layer of protection against exploits in Rich OS
- Linux kernel: 265 vulnerabilities in 2019

Isolated environment

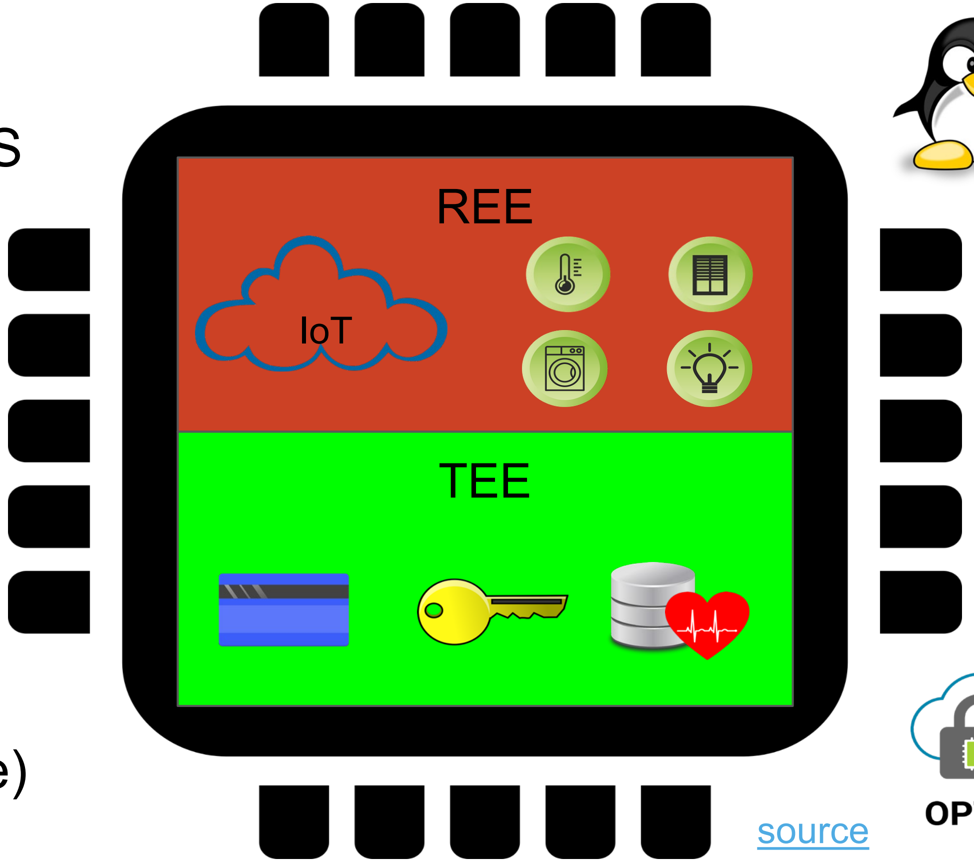
- Code/Data confidentiality & integrity

Runs alongside a REE (Linux distro, Android, etc.)

- Rich Execution Environment
- Provides services to apps on REE

Policy in software, enforced by hardware

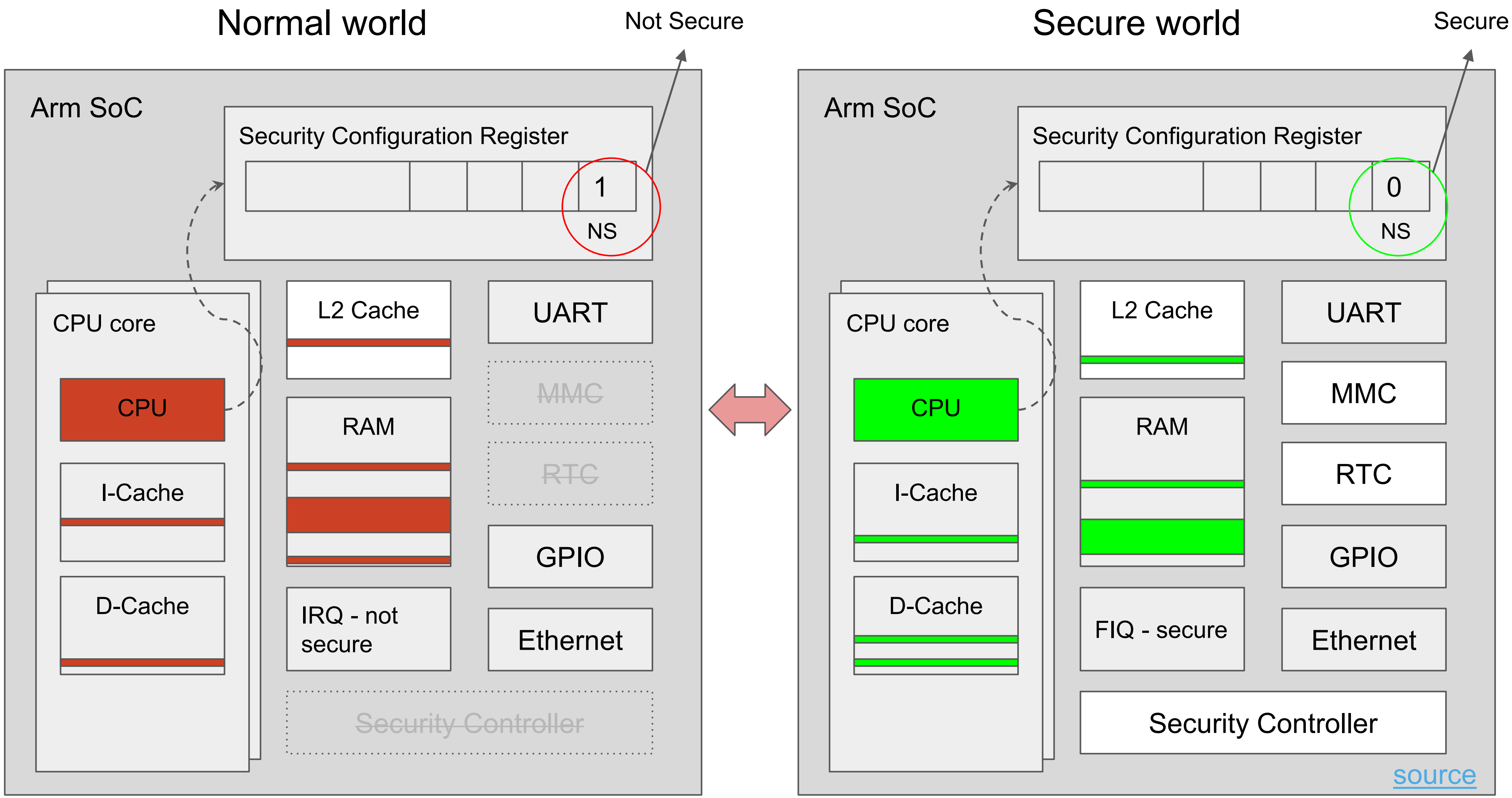
- Arm TrustZone (memory/IO protection based on state)



source

OPTEE

Simplified Hardware View of Arm TrustZone



TEE Use Cases

- Secure key handling
  - Potentially replace dedicated security chips (e.g.: HSM/TPM) and still perform secure key storage, signing, attestation, and more
- Protect Intellectual Property by providing data decryption for DRM purposes
- Protect sensitive data processing/algorithms with a Trusted User Interface/Application
  - Payment info, fingerprint authentication, and more
- Hardware level data security
  - Replay Protected Memory Blocks (RPMB) on eMMC
- Hardware level memory security
  - DDR firewalls (via TrustZone Address Space Controller [TZASC])

OP-TEE

Overview

SECURE CONNECTIONS
FOR A SMARTER WORLD

Choosing Secure World OS

Global Platform TEE API and Framework Spec

- TEE Client API
- TEE Internal Core API, etc.

Commercial/proprietary and open source

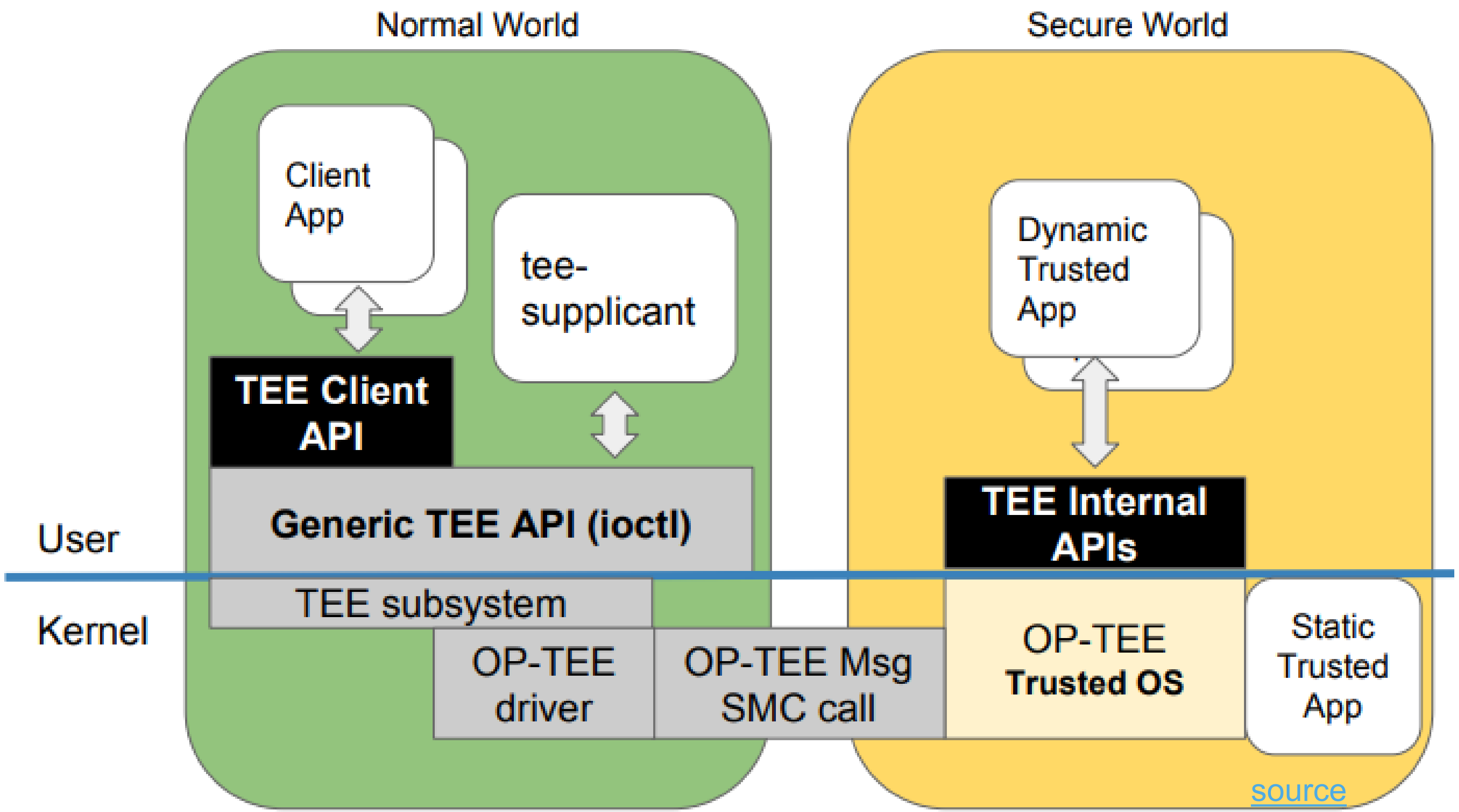
- Features, support

Open-source Portable TEE (OP-TEE)

- 50+ platforms/SoCs supported
- Global Platform TEE specification compliant
- OS, client: BSD 2-clause license, Linux driver, test suite: GPLv2
- Maintained by Linaro

OP-TEE Architecture

- Provided by open source ecosystem:
  - Operating Systems
  - Linux Driver
  - OP-TEE Client
- Product team develops:
  - Trusted Application
    - Runs from OP-TEE
  - Client Application
    - Runs from Linux/REE

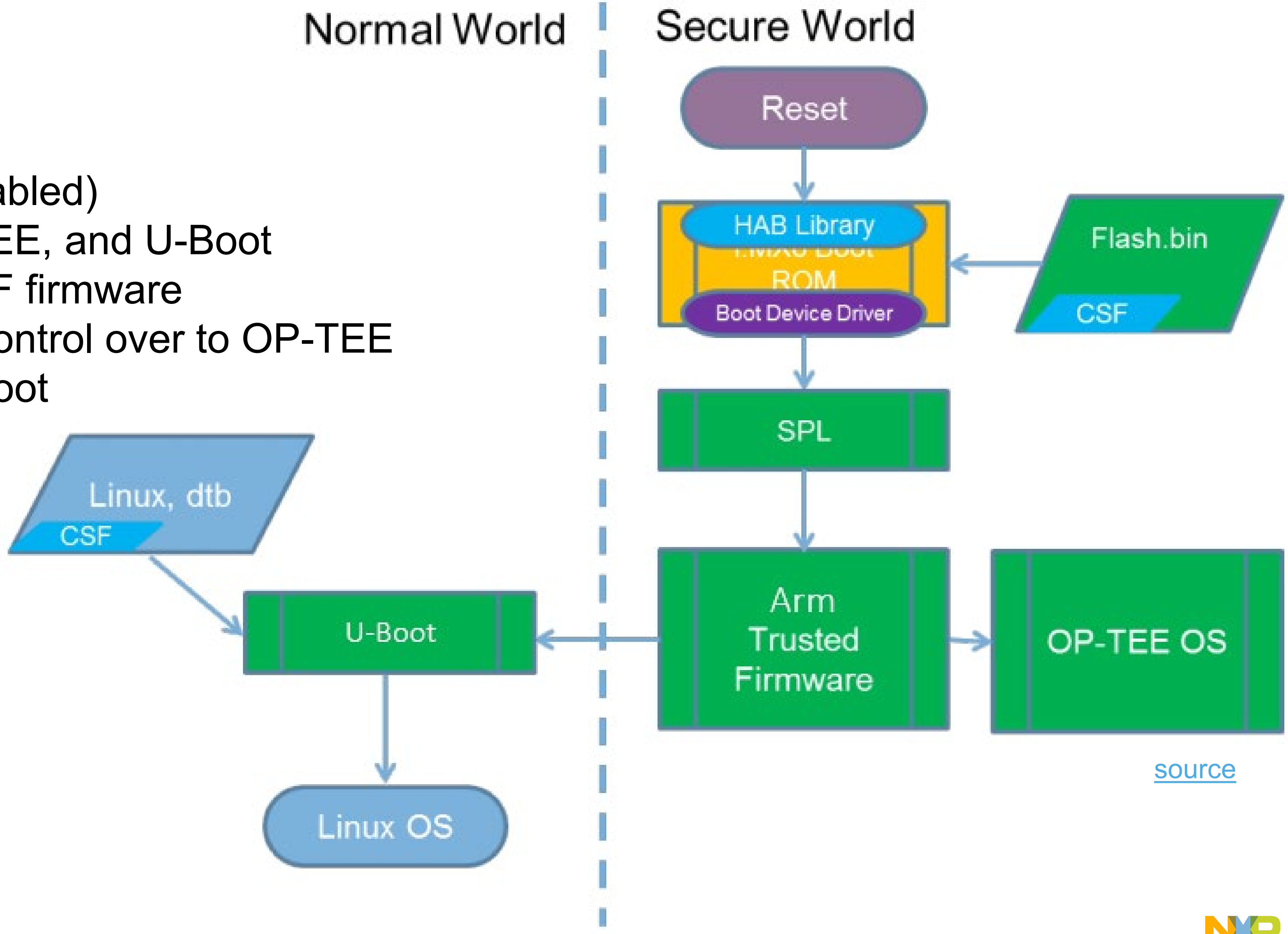


Arm Trusted Firmware

- Reference implementation of secure world software for Arm architectures (aarch32 and aarch64)
- On i.MX 8M, ATF (alongside the SCU) currently partitions non-secure resources for the OS partition before launching OP-TEE
- ATF also provides the secure monitor code to manage the switch between secure and non-secure world
- On the i.MX 6/7 platforms, by default (without OP-TEE), U-Boot and Linux run in a secure world context
- On the i.MX 8M/MN/Mini, the ATF switches U-Boot and Linux into a non-secure context by default

i.MX 8M OP-TEE Boot Flow

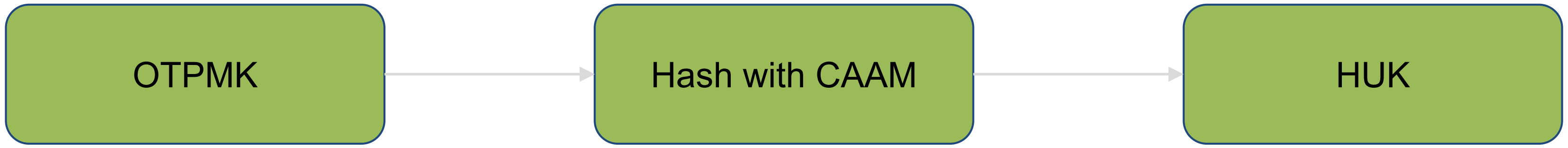
1. Power on
2. HAB verification (if enabled)
3. SPL loads ATF, OP-TEE, and U-Boot
4. SPL then jumps to ATF firmware
5. ATF firmware hands control over to OP-TEE
6. OP-TEE jumps to U-Boot



source

OP-TEE: Security with i.MX Platform

- i.MX's CAAM (Cryptographic Accelerator and Assurance Module) can be utilized for
  - Seeding and/or generating random numbers with OP-TEE
  - Creating separate hardware unique keys (HUKs) for secure/normal world by hashing the i.MX's one-time programmable master key (OTPMK)
    - HUK can then be used for various security applications without need for separate TPM/HSM



* Generated hash differs between normal/secure world

- Note: CAAM is available on many, but not all i.MX processors. It is only used if available.

Memory Protection

SECURE CONNECTIONS
FOR A SMARTER WORLD

TZASC380 – RAM Protection

- TrustZone Address Space Controller
- IP developed by Arm, designed to provide configurable protection over DRAM memory space.
  - Supports 16 independent address regions.
  - Access controls are independently programmable for each address region.
  - Sensitive registers may be locked.
  - Host interrupt may be programmed to signal attempted access control violations.
- 32 MB of the RAM space are allocated to OP-TEE
  - 28 MB is mapped by the TZASC as secure (OP-TEE RAM) i.e. ***[0xFE000000 to 0xFFC00000]***
  - 4 MB is mapped as non-secure (shared memory) i.e. ***[0xFFC00000 to 0xFFFFFFFF]***
  - On i.MX 8, SCFW divides/partitions these resources.
  - Note: In OP-TEE 3.7 (and certain i.MX 8 platforms) base addresses are shifted to be within the first 1GB of DDR
- Depending on OP-TEE version, these are defined inside
  - optee-os/core/arch/arm/plat-imx/tzasc.c
  - optee-os/core/arch/arm/plat-imx/drivers/tzc380.c

Peripheral Security

- Arm TrustZone can also be configured to secure peripheral address spaces (UART, I2C, SPI, etc.) if desired
  - Peripheral control must then be performed inside the secure world (typically with an authorized API call from the normal world)

i.MX 8 DDR Example Memory Regions

- Many of these regions are also marked as reserved inside the Linux DTB, such that Linux will not (and cannot) allocate them for use

0x80000000 to 0x8001FFFF	Secure ATF	Reserved by ATF
0x80020000 to 0x801FFFFF	Non-secure OS	Reserved by UBoot
0x80200000 to 0x87FFFFFF	Non-secure OS	-
0x88000000 to 0x887FFFFF	M4_0	Reserved by SCFW/U-Boot for Cortex-M4
0x88800000 to 0x8FFFFFFF	M4_1	Reserved by SCFW/U-Boot for Cortex-M4
0x90000000 to 0xFDFFFFFF	Non-secure OS	-
0xFE000000 to 0xFFBFFFFF	Secure ATF	Reserved by ATF for OPTEE
0xFFC00000 to 0xFFFFFFFF	Non-secure OS	-
0x880000000 to 0x8C0000000	Non-secure OS	-

- Note: In systems which do not utilize the maximal supported DDR address space the TZASC must be configured to protect all aliased regions as well
- Only an example: depending on processor and OP-TEE version 3.7, these addresses may have changed

Verifying Memory Protection

- Linux can’t read OP-TEE memory

```
root@imx8qxp:~# devmem2 0xFFC00000
    Memory mapped at address 0xffffa24a0000.
    Read at address  0xFFC00000 (0xffffa24a0000): 0x00000000
root@imx8qxp:~# devmem2 0xFE000000
    Memory mapped at address 0xffff98656000.
    Bus error
```

Trusted Applications (TA)

Overview

SECURE CONNECTIONS FOR A SMARTER WORLD

Trusted Applications

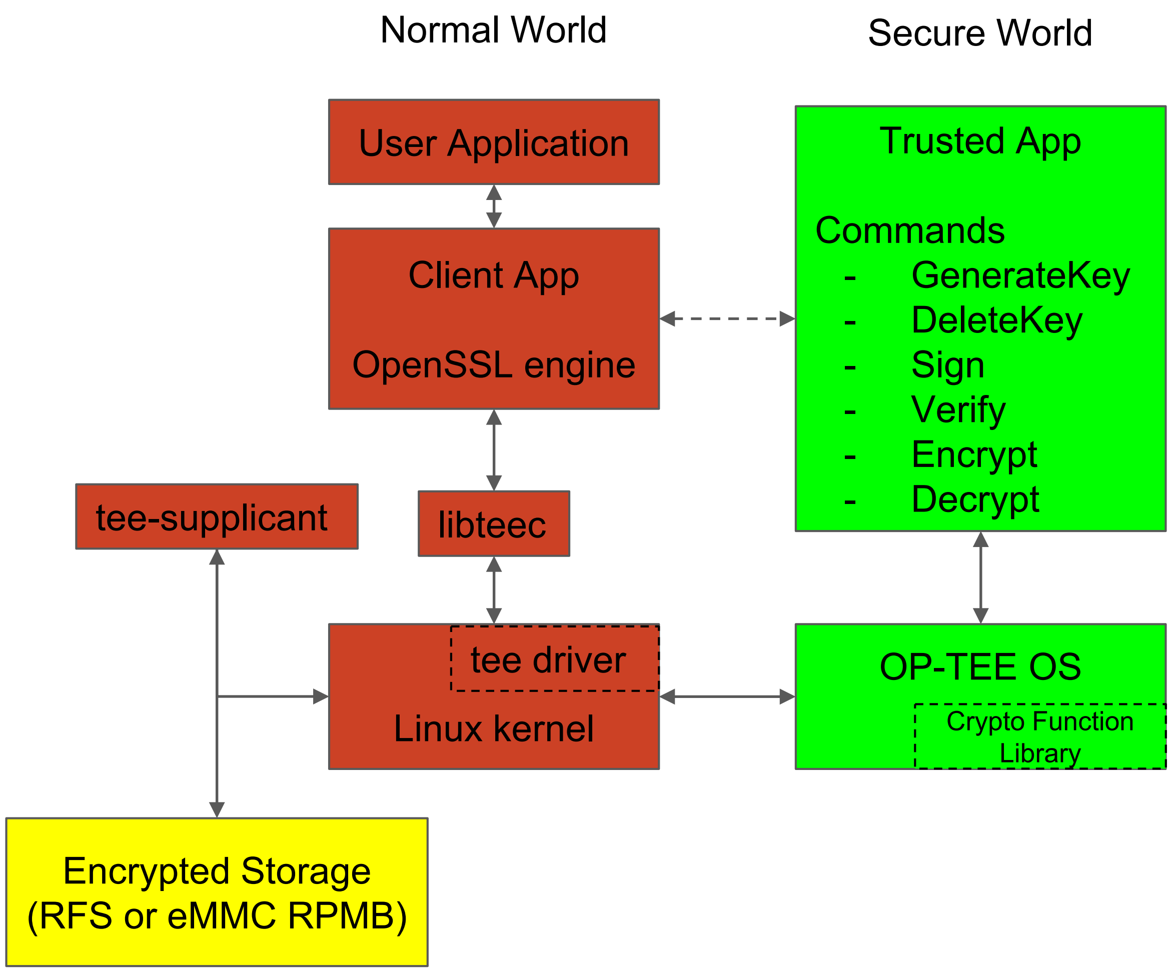
Secure World Trusted App

- Cryptographic functions (CAAM accelerated, mbed TLS library)
- Secure data storage
  - AES GCM encrypted file in REE (/data/tee)
  - eMMC RPMB

Linux User Space Client App

- Shared memory

Example TA Architecture



Anatomy of CA and TA

Hello world example found here

Client Application

TEEC_InitializeContext

TEEC_OpenSession

TEEC_InvokeCommand

TEEC_CloseSession

TEEC_FinalizeContext

Trusted Application

TA_CreateEntryPoint

TA_OpenSessionEntryPoint

TA_InvokeCommandEntryPoint

TA_CloseSessionEntryPoint

TA_DestroyEntryPoint

Trusted Application – Build Environment

- To build a trusted application, you'll need to setup the TA dev kit
  - Included as part of optee_os (optee_os/blob/master/ta/mk/ta_dev_kit.mk)
- The trusted application then uses the ta_dev_kit.mk path while building
  - For the hello_world example, source/Makefile are located at optee_examples/hello_world/ta
  - Once built, this produces a UUID filename that is used to load the TA when you start the host application from the REE
- The host application is also similarly built:
  - For the hello_world example, source/Makefile are located at optee_examples/hello_world/host
  - Once built, this produces an Arm executable that can be run from the REE/Linux (optee_example_hello_world)

- When building these externally, there are many environmental variables that must be set up manually. Yocto manages most of this for you… so use it instead!

Trusted Application – Executing

- Once both are built, you end up with an application pair such as:

Application name	UUID
optee_example_hello_world	8aaaf200-2450-11e4-abe2-0002a5d5c51b

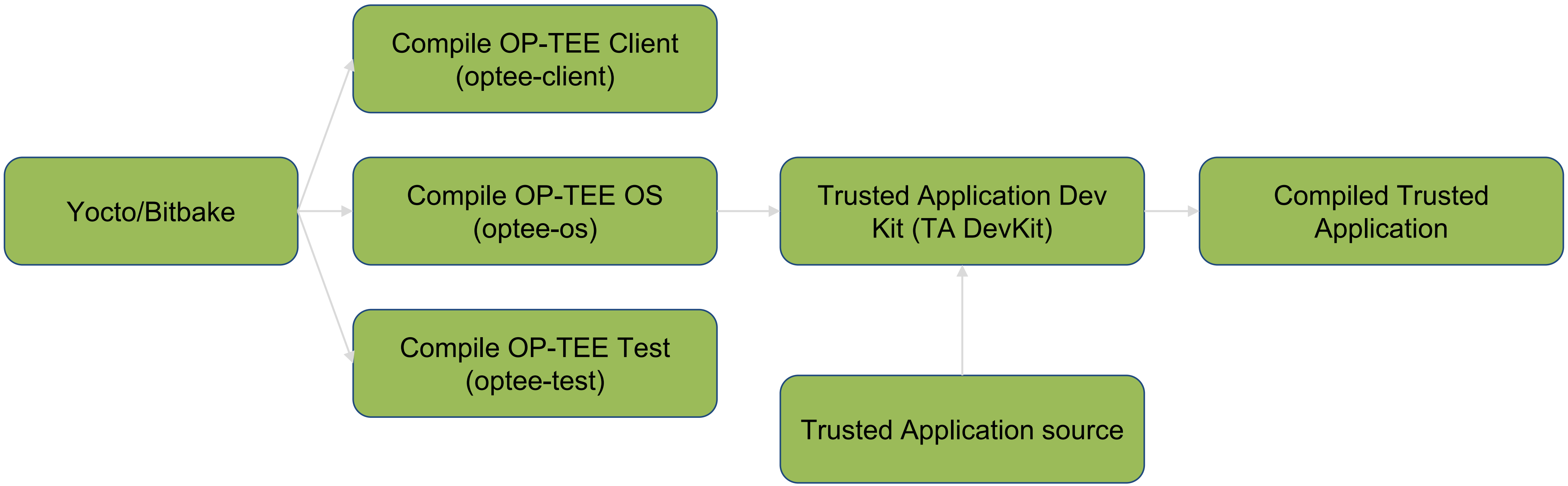
- From the REE/Linux, this can then be run:
  - root@imx8: optee_example_hello_world
- OP-TEE then knows what the corresponding TA UUID is and will load/execute it
  - The UUID application is generally stored at /lib/optee_armtz/*.ta on the REE (Linux)

Enabling & Testing

via Yocto

SECURE CONNECTIONS
FOR A SMARTER WORLD

Trusted Application Compilation Flow



- Also worth mentioning, only a properly signed TA will execute at runtime...
  - Public portion of keypair is compiled into OP-TEE
  - TA is then signed with private key after compilation
  - OP-TEE verifies TA signature when loading/executing

i.MX 8 OP-TEE Device Tree Configuration

- Automatically added into device tree by U-Boot or ATF on i.MX 8:
- Enables OP-TEE driver (linux/drivers/tee/optee/core.c)

```
firmware {
    optee {
        compatible = "linaro,optee-tz";
        method = "smc";
    };
};
```

Building with OP-TEE

Yocto

1)

```
MACHINE_FEATURES += "optee"
DISTRO_FEATURES += "optee"
IMAGE_INSTALL += "optee-test optee-os optee-client"
```

U-Boot (i.MX 6/7)

2)

```
CONFIG_IMX_OPTEE=y
```

* Does not appear to be necessary on i.MX 8 during testing

Linux (i.MX 6/7/8)

3)

```
CONFIG_OPTEE=y
```

4) → bitbake core-image-minimal

OP-TEE Test Suite: XTest

- Once built, confirm OP-TEE's functionality with the test suite
  - XTest runs various operations and checks for correct functionality
- Running the test suite:

```
root@imx8:~# ls /dev/tee*
    /dev/tee0  /dev/teepriv0
root@imx8:~# xtest
    …
    16099 subtests of which 0 failed
    74 test cases of which 0 failed
    0 test case was skipped
    TEE test application done!
```

Enhanced OpenSSL

Why?

- Abstract method to utilize OP-TEE OS
  - Can use OpenSSL instead of writing custom code
- Hardware accelerated cryptographic operations
- Additional layer of security for key storage
- Following operations are supported:
  - RSA/ECC key-pair generation
  - RSA/ECC key-pair import
  - SHA/MD5 hash digest generation
  - RSA PKCS decryption
  - RSA/ECC signature generation

Enhanced OpenSSL Block Diagram

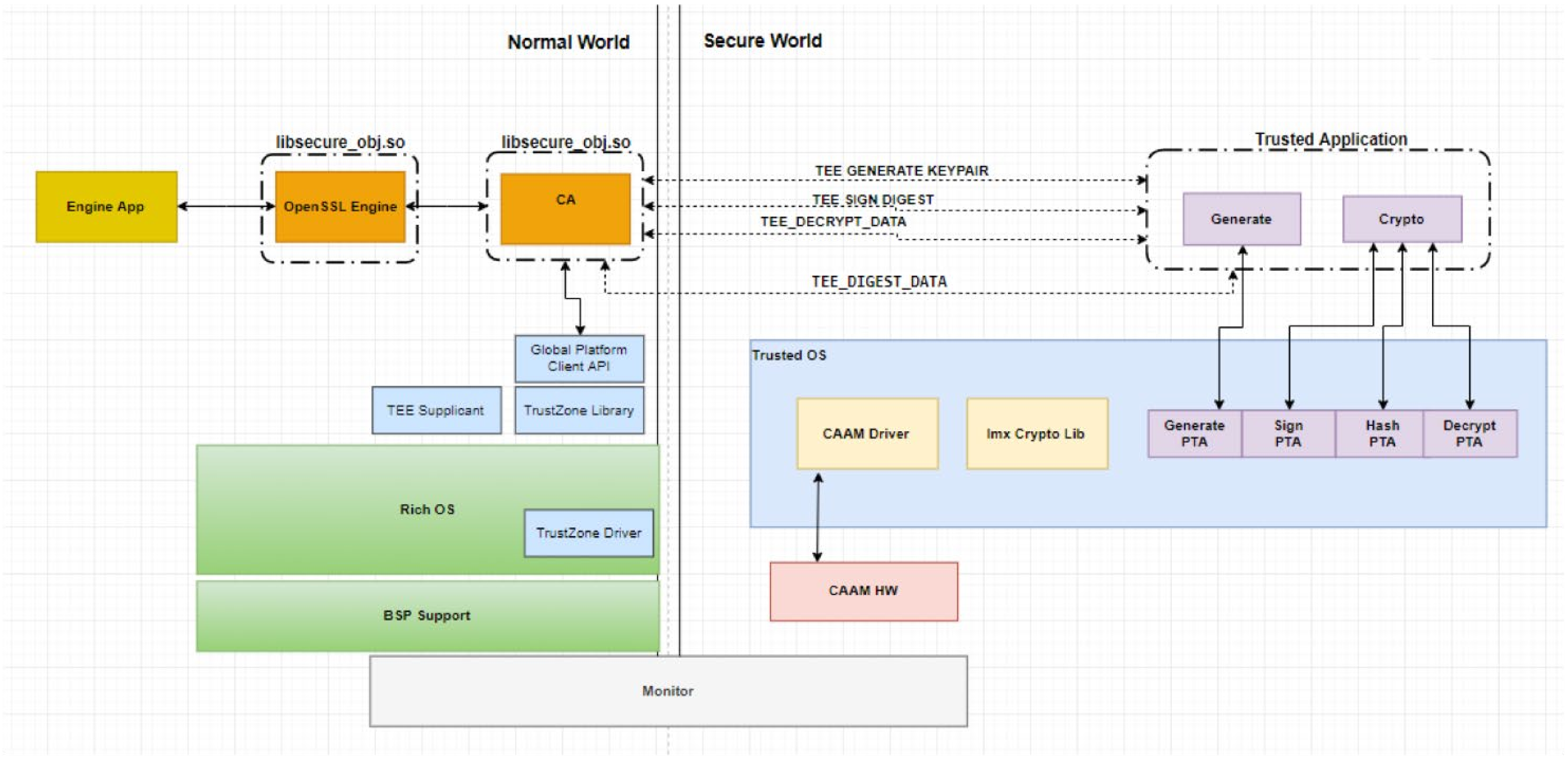


Figure 1.

source

Example: Sign and Verify Data through HSM/TEE

1. Set OpenSSL loading information for enhanced engine (can also modify openssl.cnf)
    - $ export IMXENGINE="-pre SOPATH:/usr/lib/libengsecureobj.so -pre ID:engsecureobj -pre LISTADD:1 -pre LOAD"
    - $ openssl engine ${IMXENGINE} -t dynamic
2. Generate a private key in the HSM with sobj_app, This will also create a fake PEM (which contains information to get required key from HSM)
    - $ sobjapp -G -m rsa-pair -s 2048 -l "rsagen2048" -i 1 -w rsa2048.pem
3. Retrieve Public Key
    - $ openssl rsa -in rsa2048.pem -pubout -out rsapub2048.pem
4. Sign data
    - $ openssl dgst -sha1 -sign rsa2048.pem -out sig.data data
5. Verify data
    - $ openssl dgst -sha1 -verify rsapub2048.pem -signature sig.data data

Other Security Considerations

SECURE CONNECTIONS
FOR A SMARTER WORLD

Other Security Considerations

- Disable JTAG (i.MX 6/7/7M)
- Setup JTAG for secured access only (i.MX 8/8X)
- Enable secure boot
  - Establish chain of trust (FIT image, SOC specific APIs)
  - Without secure boot, OP-TEE cannot be truly secure!
    - (as it could be replaced with a modified version)
- Review non-secure world permissions
  - Review all bootloaders prior to OP-TEE
- Review use of keys
  - Secure storage requires unique hardware key
- Disable login prompts
- Disable username/password access via SSH. Require key pairs if you must use SSH

Other Security Considerations

- Check your included open source software packages for Common Vulnerabilities and Exploitations (CVEs) on the national vulnerability database
- A typical embedded system has hundreds or thousands of these packages which must be checked. There are tools which can help!
  - Vigiles Vulnerability Management
- Even OP-TEE has CVEs

Takeaways

- TEE provides inexpensive additional security layer
- OP-TEE makes adoption of TEE easier
- Make security requirements part of your product requirements from day 1
- If needed, leverage assistance of experienced security development teams from NXP and Timesys:
  - Product security design
  - Configuration and implementation of needed security features
  - Additional security documentation
  - Security verification
  - Compliance alignment
- Start with initial non-binding conversation

Additional Resources

More info can be found on OP-TEE upstream repositories:

- https://github.com/OP-TEE/optee_os/tree/master/documentation
- Upstream repositories are available at: https://github.com/OP-TEE/
- OP-TEE website: https://www.op-tee.org/

Enhanced OpenSSL

- http://source.codeaurora.org/external/imxsupport/imx_sec_apps

Upcoming Webinars

SECURE CONNECTIONS
FOR A SMARTER WORLD

In-depth Dive

- **Linux Kernel Security:** Overview of Security Features and Hardening
- **Security Hardening:** Protecting Your Embedded Linux Device from the Risk of Being Compromised
- **Designing OTA Updates:** An Integral Part of a Secure System

Previous Webinars

SECURE CONNECTIONS
FOR A SMARTER WORLD

Previous Webinars

Secure By Design Series

- Securing Embedded Linux Devices: Pitfalls to Avoid
- Software integrity and data confidentiality: Establishing secure boot and chain of trust on i.MX processors

Stay Secure (Vigiles) Series

- Software Security Management: Cutting through the vulnerability storm with NXP Vigiles
- BSP security maintenance: Best practices for vulnerability monitoring & remediation
- Full Life Cycle Security Maintenance of Embedded Linux BSPs
- Best practices for triaging Common Vulnerabilities & Exposures (CVEs) in embedded systems

For More Information and to Become More Secure

GOLD PARTNER NXP

Timesys is an embedded Linux security expert and NXP Gold Partner.
To discuss your project, please contact us at sales@timesys.com

Use this link to go to Services for securing your device

Thank You!

Q&A

SECURE CONNECTIONS
FOR A SMARTER WORLD