OP-TEEワークショップ

セコム株式会社IS研究所

宮澤慎一

2024/05/10

ARMの誕生からTrustZone誕生まで

ARMの歴史的な経緯に関する主な参考文献：
Dingee, Don, and D. Nenni. "Mobile Unleashed: The Origin and Evolution of ARM Processors in Our Devices.” SemiWiki LLC (2015).

ARMのバージョンと実装名などの関係

ARM アーキテクチャ仕様	ARM core 実装	SoC	ハードウェア プラットフォーム
ARMv6	ARM1176JZF-S	Broadcom BCM2835	Raspberry Pi 1
ARMv8-A	ARM Cortex-A72	Broadcom BCM2711	Raspberry Pi 4
ARMv7-A	ARM Cortex-A9	Apple A5	iPhone 4S
ARMv7-A	Apple Swift	Apple A6	iPhone 5
ARMv8-A	重：Apple Avalanche 軽：Apple Blizzard	Apple A15	iPhone 13

https://www.raspberrypi.com/documentation/computers/processors.html
https://phonedb.net/index.php?m=processor&id=282&c=apple_a5_apl0498__s5l8940
https://phonedb.net/index.php?m=processor&id=356&c=apple_a6_apl0598__s5l8950x
https://phonedb.net/index.php?m=processor&id=871&c=apple_a15_bionic_apl1007__apl1w07__t8110

ARMチップが誕生した理由

- **ACORN社**
  - 大ヒット家庭用コンピューターBBC Micro(1981-) のCPU(MOS6502)の後継がなかなか発表されなかった
  - IntelやMotrallaのCPUを試したが割り込み処理が遅く、ACORN社の希望の速度に達成しない
- **低価格で高速な自社CPUを目指す**
  - 低価格 = チップ面積が小さい
  - MOS 6502: 3,510
  - Intel386: 275,000
  - ARM1: 25,000
  - ARM2: 27,000
- **Sophie Wilson (Robin Wilson)がARMを設計**
  - UCバークレーのRISC論文(1980)を読んだことがきっかけ。
- **VLSI社が製造**

Dingee, Don, and D. Nenni. "Mobile Unleashed: The Origin and Evolution of ARM Processors in Our Devices.” LLC (2015).
Patterson, David A., and David R. Ditzel. "The case for the reduced instruction set computer." ACM SIGARCH Computer Architecture News 8.6 (1980): 25-33.

ARMのCPUモードやセキュリティ拡張の歴史

	年	ISA ver.	プロセッサ	追加機能（CPUモードやセキュリティ拡張）
Acorn	1985	ARMv1	ARM1(試作)	User Mode, FIQ Mode, IRQ Mode, Supervisor Mode
	1986	ARMv2	ARM2 ARM3	MEMC搭載
ARM Ltd.	1991	ARMv3	ARM610 ARM700	Abort Mode, Undefined Mode, Cache搭載
	1993	ARMv4	ARM710T ARM810 ARM9TDMI	System Mode
	1997	ARMv5	ARM926EJ-S ARM1022E	Java拡張(2000)
	2002	ARMv6	ARM1176JZ(F)-S SC000 Cortex-M0	Monitor Mode, **TrustZone拡張(2003)**
	2004	ARMv7	Cortex-A5 - A9 Cortex-M3,4 Cortex-R4,5,6	Hyp Mode (2010)
	2011	ARMv8	Cortex-A5x	64bit
	2014	ARMv8.1		Virtualization Host 拡張
	2017	ARMv8.4		Secure Virtualization拡張
	2021	ARMv9		**ARM CCA発表**

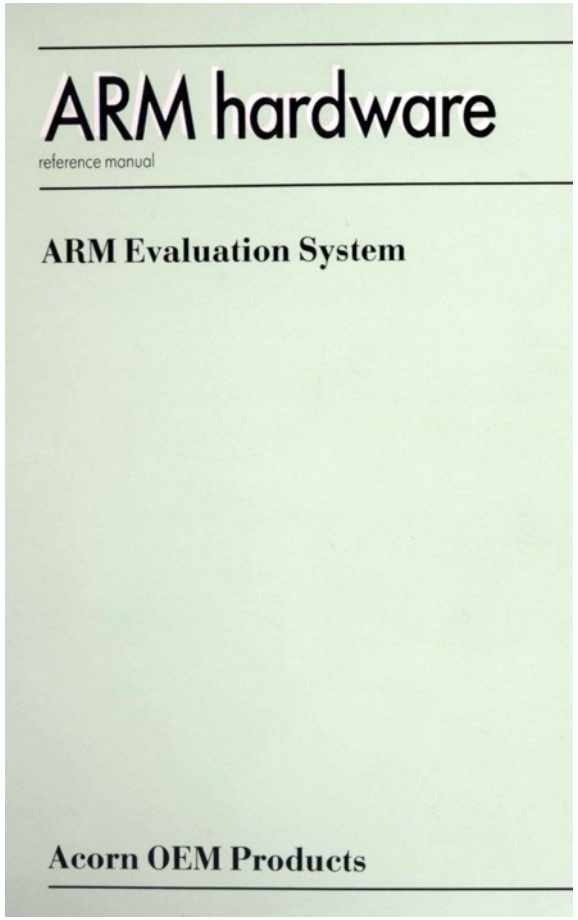
参考文献は次ページ

ARMのCPUモードやセキュリティ拡張の歴史

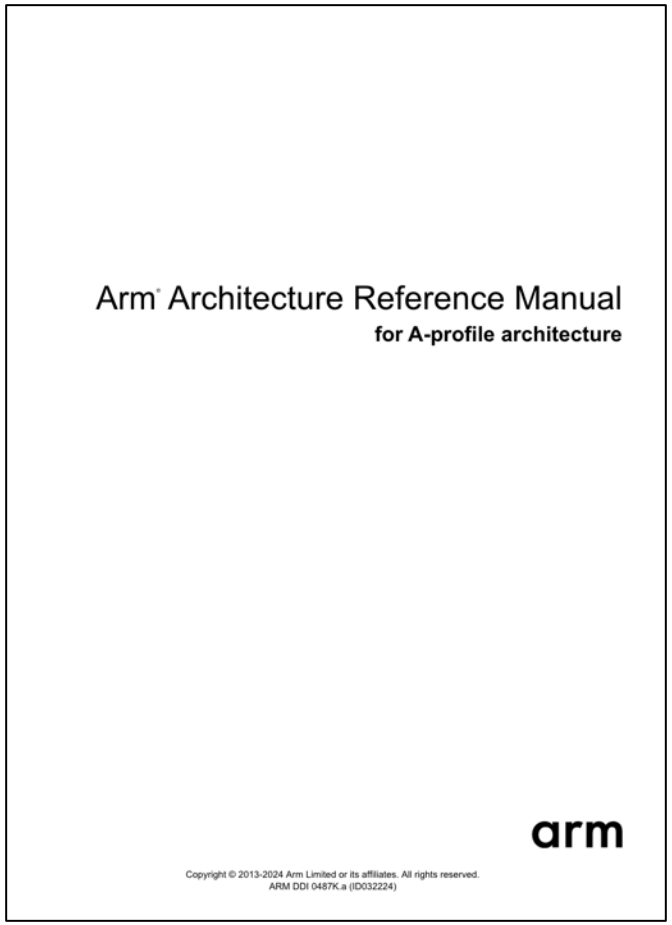
前ページの表に関する参考文献

Armv1 https://en.wikichip.org/wiki/acorn/microarchitectures/arm1
Armv2 https://www.itmedia.co.jp/news/articles/2006/08/news058_2.html
Armv3 https://en.wikichip.org/wiki/arm_holdings/microarchitectures/arm6
System Mode https://developer.arm.com/documentation/dui0471/g/Handling-Processor-Exceptions/Use-of-System-mode-for-exception-handlin
Java拡張 https://ascii.jp/elem/000/000/317/317686/
TrustZone https://pc.watch.impress.co.jp/docs/2003/1022/arm.htm
TrustZone https://www.sigmobile.org/pubs/getmobile/articles/Vol22Issue3_1.pdf
その他 https://en.wikipedia.org/wiki/ARM_architecture_family

Reference Manual



ARM1 (1985) **56**ページ
ARM hardware reference manual



ARMv8,v9-A (2024) **14,777**ページ
Arm Architecture Reference Manual
for A-profile architecture

Reference Manual

Processor mode bits (m1,m0)

These are output signals which are the inverses of the internal status bits indicating the processor operation mode.

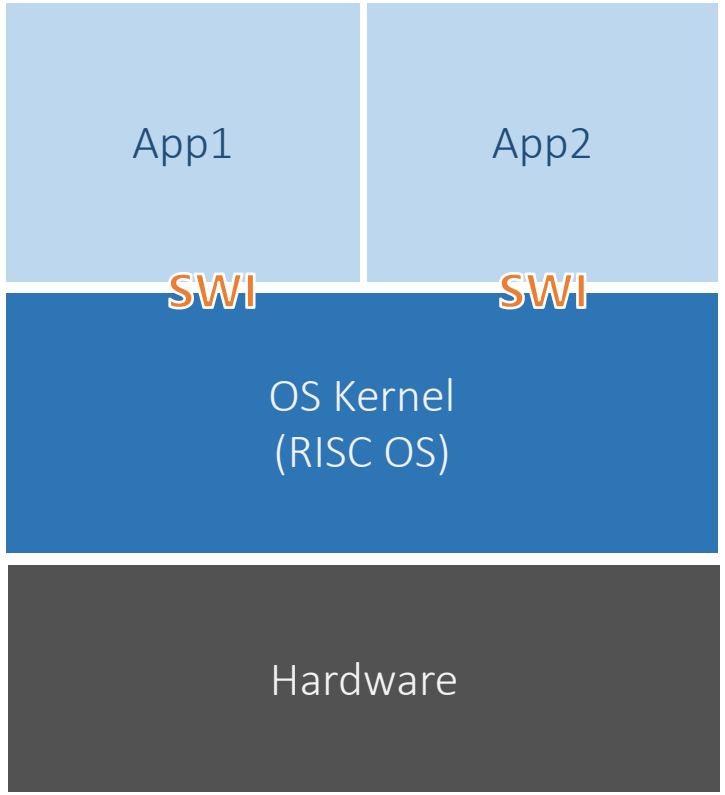
m1	m0	mode
0	0	supervisor
0	1	irq
1	0	fiq
1	1	user

Note: this table gives the values at the pins; the internal bits are inverted with respect to this table.

ARM hardware reference manual

Arm Architecture Reference Manual for A-profile architecture

ARM v2 (1986 -)



RISC OS OPEN

Safeguarding the past, present and future of RISC OS for **everyone**

Home | News | Downloads | Bugs | Bounties | Forum | Documents

OS SWI Calls

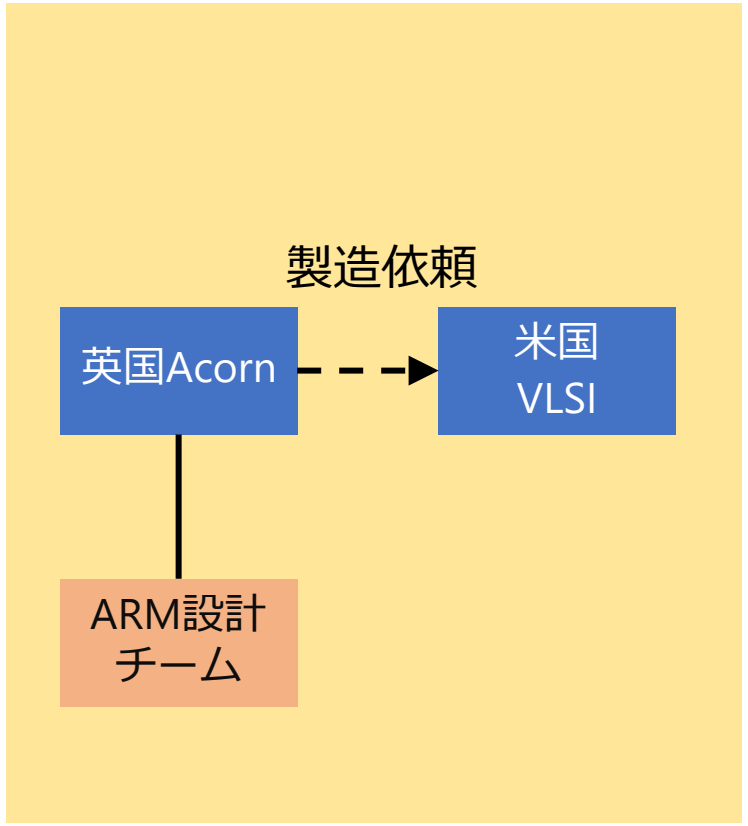
Programmer's Reference Manuals
» OS SWI Calls

- OS_AbortTrap
- OS_AddCallBack
- OS_AddToVector
- OS_AMBControl
- OS_Args
- OS_BGet
- OS_BinaryToDecimal
- OS_BPut
- OS_BreakCtrl
- OS_BreakPt
- OS_Byte
- OS_CallAfter
- OS_CallASWI
- OS_CallASWIR12
- OS_CallAVector
- OS_CallBack
- OS_CallEvery
- OS_ChangedBox

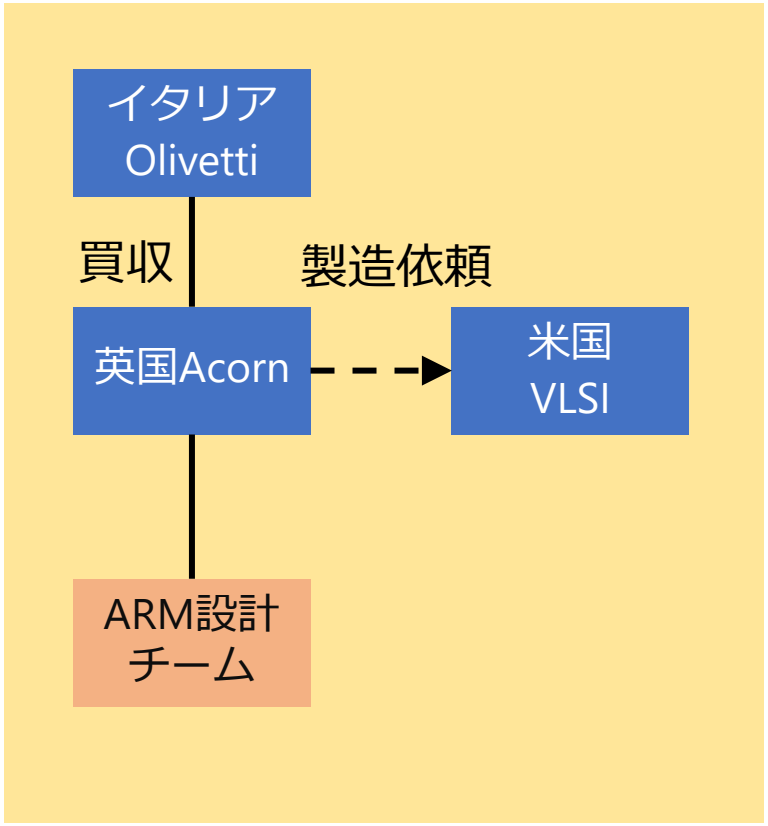
OS SWI Calls in Library (riscosopen.org)

システムコール＝ソフトウェア割り込み＝SoftWare Interrupt Call

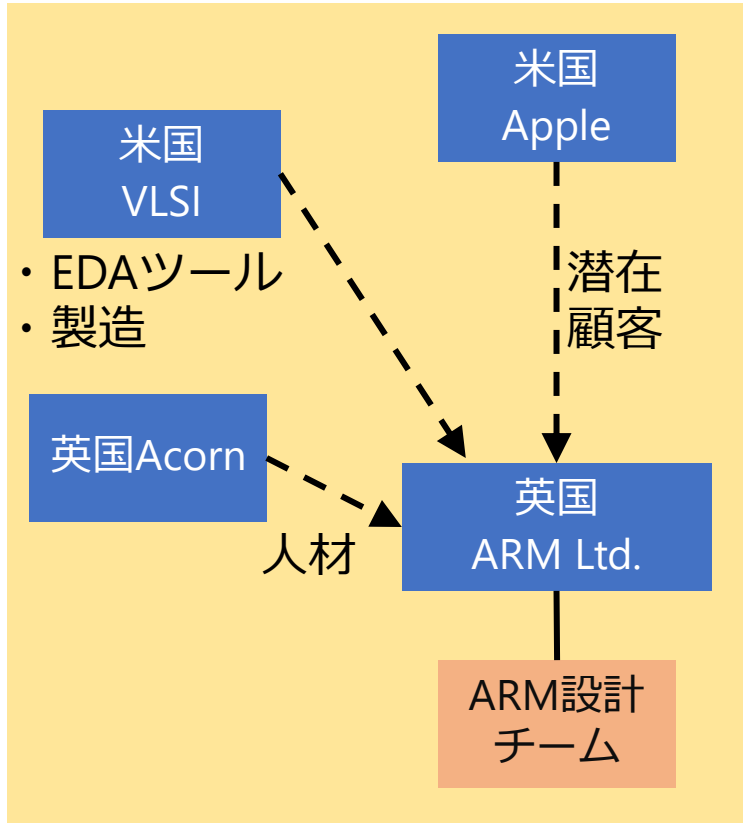
ARM設立 (1990-)



(1978-) 1983-1985



1985-1998



1990-

Acorn RISC Machine

Advanced RISC Machine

Dingee, Don, and D. Nenni. "Mobile Unleashed: The Origin and Evolution of ARM Processors in Our Devices." LLC (2015).

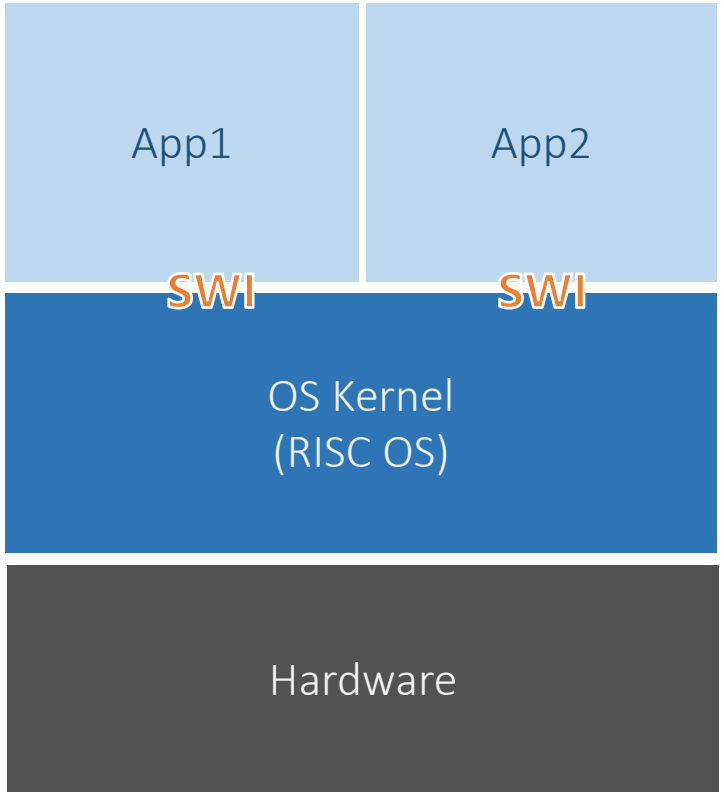
ARM設立 (1990-)

- **Larry Tesler**
  - Xerox → Apple→Newtonプロジェクトマネージャ
  - Newtonがうまくいっていない時にARM2の性能に衝撃を受けた。
- **Robin Keith Saxby**
  - Motorola → European Chip (Atmel) →ARM初代CEO
  - Larry Teslerにスカウトされ、Acorn社の技術コンサルになった後、ARM初代CEO
  - デスクトップ向けCPUになる可能性もあったARM CPUを、組み込み向けで展開する方針をとって成功をおさめた

Dingee, Don, and D. Nenni. "Mobile Unleashed: The Origin and Evolution of ARM Processors in Our Devices.” LLC (2015).

ARM v3 (1991 -)

"Acorn Archimedes" by PaulVernon1974 is licensed under CC0 1.0 Universal



RISC OS OPEN

Safeguarding the past, present and future of RISC OS for **everyone**

Home | News | Downloads | Bugs | Bounties | Forum | Documents

OS SWI Calls

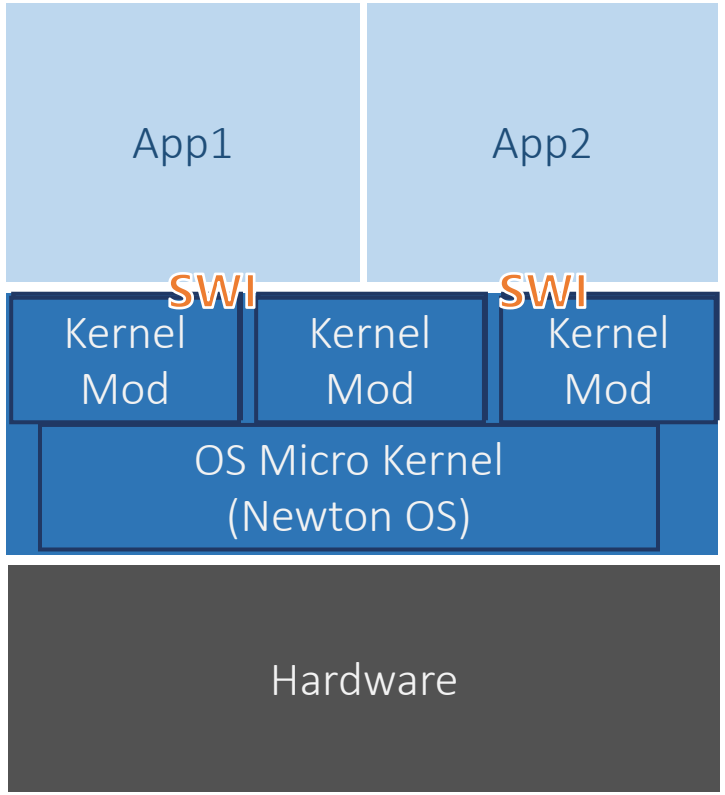
Programmer's Reference Manuals

» OS SWI Calls

- OS_AbortTrap
- OS_AddCallBack
- OS_AddToVector
- OS_AMBControl
- OS_Args
- OS_BGet
- OS_BinaryToDecimal
- OS_BPut
- OS_BreakCtrl
- OS_BreakPt
- OS_Byte
- OS_CallAfter
- OS_CallASWI
- OS_CallASWIR12
- OS_CallAVector
- OS_CallBack
- OS_CallEvery
- OS_ChangedBox

OS SWI Calls in Library (riscosopen.org)

ARM v3 (1991 -)




```
14   #define kGetPortSWI
15   #define kPortSendSWI
16   #define kPortReceiveSWI
17   #define kEnterAtomicSWI
18   #define kExitAtomicSWI
19   #define kGenericSWI
20   #define k06SWI
21   #define k07SWI
22   #define kFlushMMUSWI
23   #define k09SWI
24   #define kVersionSWI
25   #define kSemOpSWI
26   #define k12SWI
```

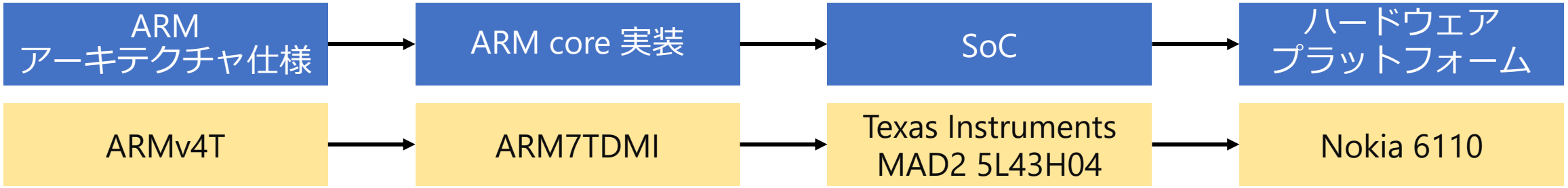
https://github.com/newtonresearch/newton-framework/blob/master/OS/SWI.h

Historic ARM Presentation to Apple Computer – 1992
https://www.youtube.com/watch?v=ZV1NdS_w4As
Apple社員向けのカンファレンスでARM社員がARMのチップ（ARM 610）を解説している動画

ARM v4 (1993 -)

ARM7TDMI (1994)
(**T**humb, jtag **D**ebug, fast **M**ultiply, embedded **I**ce)

- **普及したARM7TDMI**
  - 携帯電話 Nokia 6110
    - 初のArm搭載GSM携帯電話。大成功を収める。
  - ポータブルゲーム機：Game Boy Advance
  - ポータブルプレーヤー：iPodシリーズ



https://www.arm.com/ja/resources/blueprint/arm-official-history
https://macforlives.wordpress.com/2008/08/20/arm7tdmi-processor/
http://www.sizecoding.org/wiki/Gameboy_Advance
https://en.wikipedia.org/wiki/ARM7
https://lpcwiki.miraheze.org/wiki/Texas_Instruments

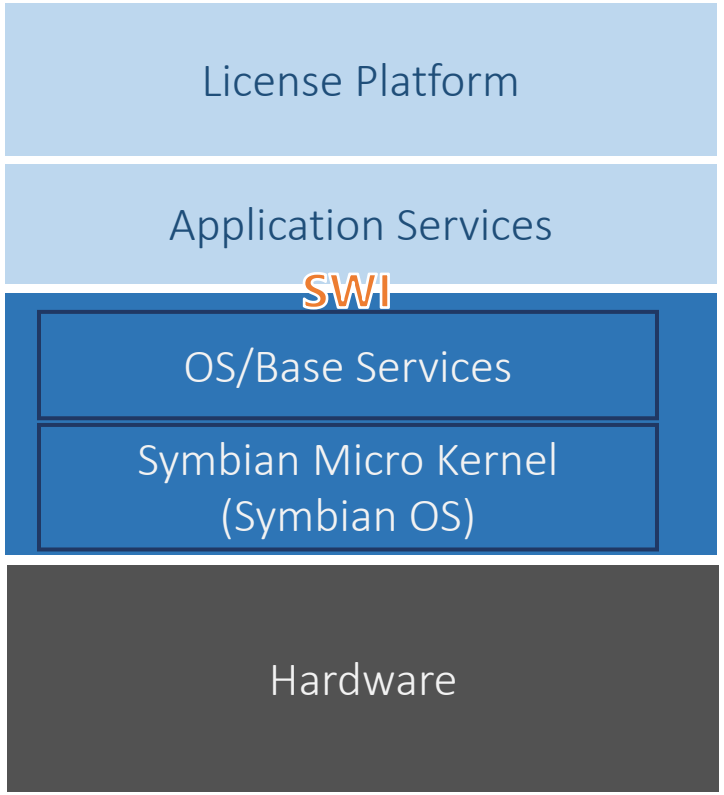
ARM v5 (1997-)

ARM926EJ-S(2001)
(**E**nhanced, **J**azzele, **S**ynthesizable)

携帯電話でJavaが普及し始める

"Nokia 6630" by Eirik Solheim is licensed under CC BY-SA 2.0

https://lpcwiki.miraheze.org/wiki/Nokia_6630



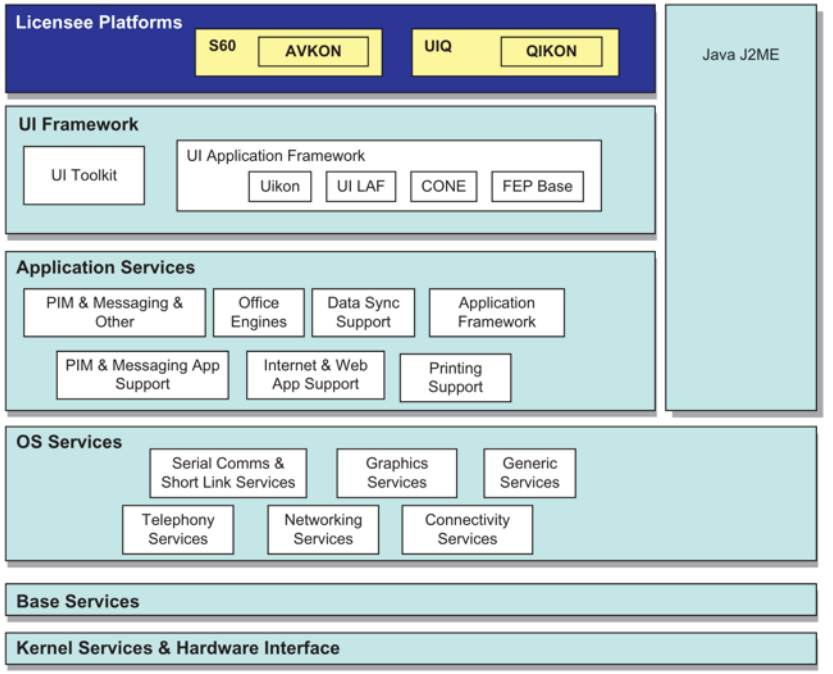


Fig. 1. Symbian Architecture. *Source:* Symbian.

West, Joel, and David Wood. "Evolving an open ecosystem: The rise and fall of the Symbian platform." Collaboration and competition in business ecosystems. Vol. 30. Emerald Group Publishing Limited, 2014. 27-67.

ARM v5 (1997-)

ARM926EJ-S(2001)
(**E**nhanced, **J**azzele, **S**ynthesizable)

前提
- モバイル端末の中には、メーカーや通信キャリアが守りたいプログラムやデータがたくさんある。
- 電波帯域制御、SIMロックプログラム、DRM（コンテンツ保護機構）

問題
- 上記のようなものが危険に晒される危険性が増加
  - ユーザーによってさまざまなアプリケーションがインストールされる
  - マルウェアが紛れ込んでしまうかもしれない
  - OSの脆弱性がゼロとは言い切れない

ベンダー側の希望
- **ユーザーとベンダーの実行環境を分けたい**

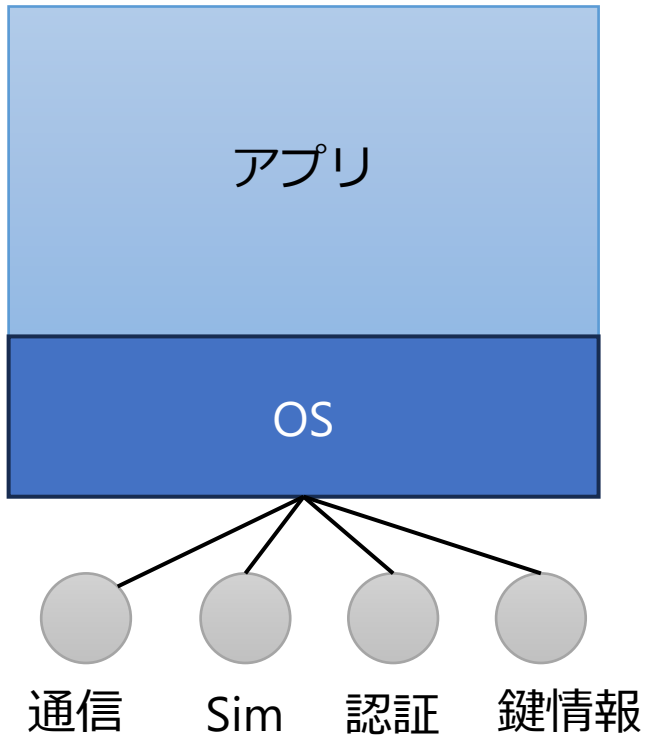
参考文献：https://blog.ssg.aalto.fi/2019/06/historical-insight-into-development-of.html

ARM v6 (2001-)　TrustZone拡張(2002)

- **課題**
  - **ユーザーとベンダーの実行環境を分けたい**
- **解決案**
  - セキュリティチップを追加する
    - コストと柔軟性に問題あり
  - 汎用CPUのチップを追加する
    - コスト、電力消費、安全性に問題あり
  - SIMに2つ目のチップとしてオフロードする
    - 性能に問題あり
- **実際の解決方法**
  - 1つのチップを論理的に分けて利用する
  - https://patents.google.com/patent/US9111097B2/ (2002)

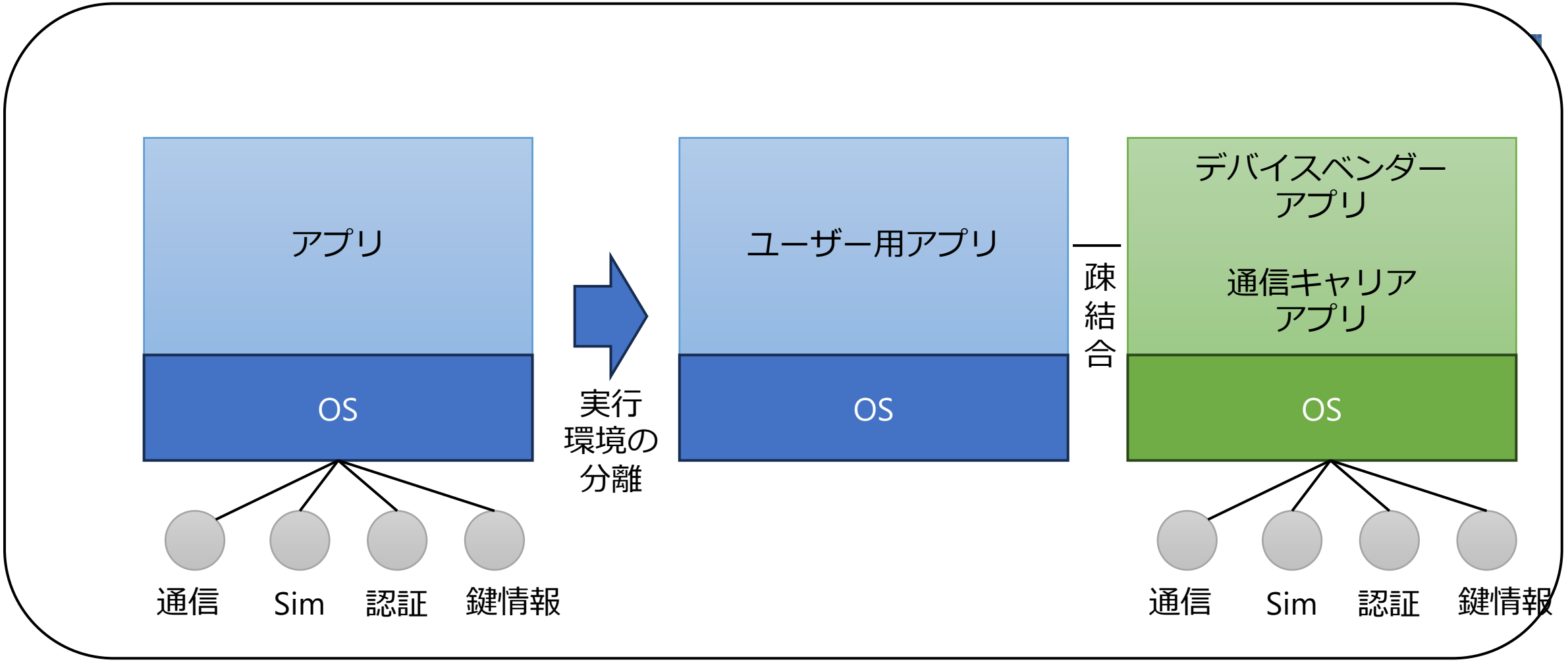
参考文献：https://blog.ssg.aalto.fi/2019/06/historical-insight-into-development-of.html

ARM v6 (2001-)　TrustZone拡張(2002)



参考文献：https://blog.ssg.aalto.fi/2019/06/historical-insight-into-development-of.html

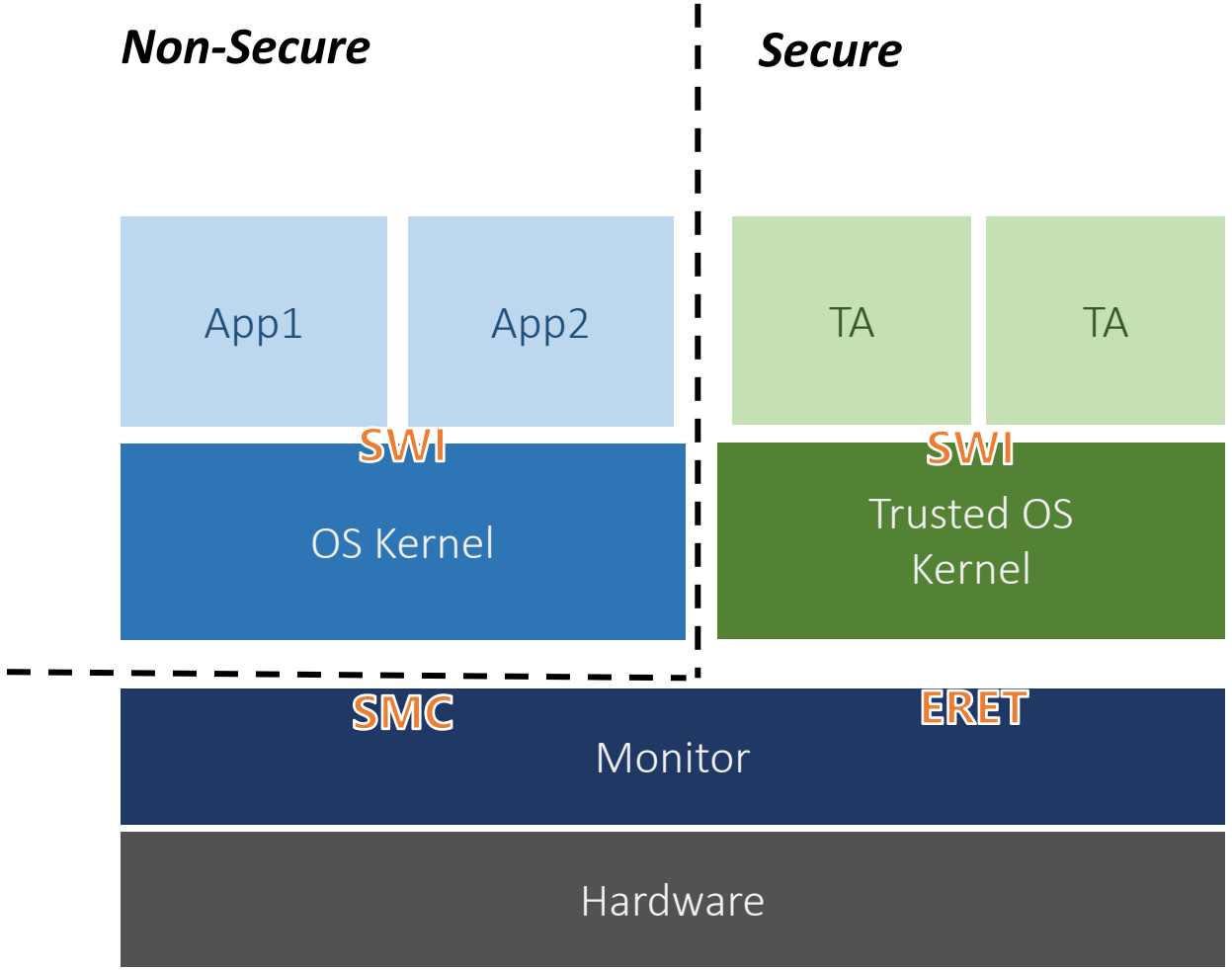
ARM v6 (2001-)　TrustZone拡張(2002)



参考文献：https://blog.ssg.aalto.fi/2019/06/historical-insight-into-development-of.html

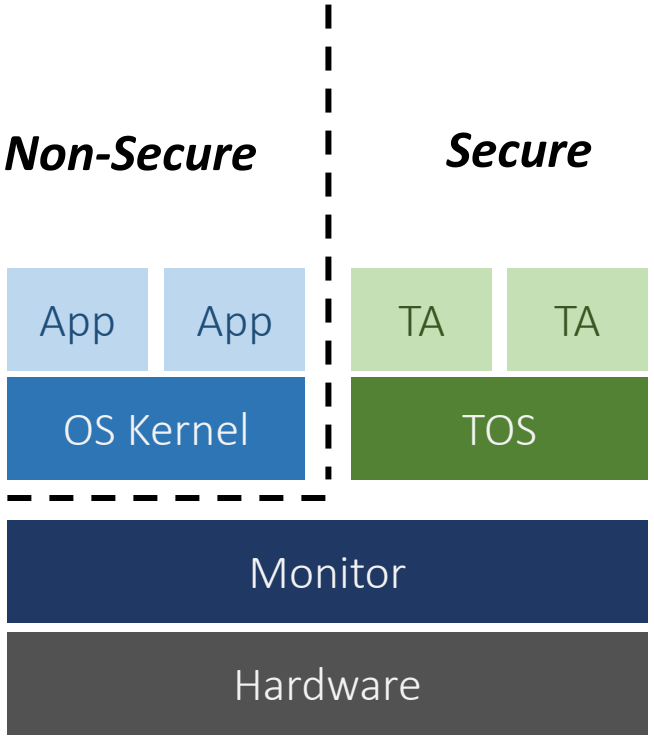
ARM TrustZone

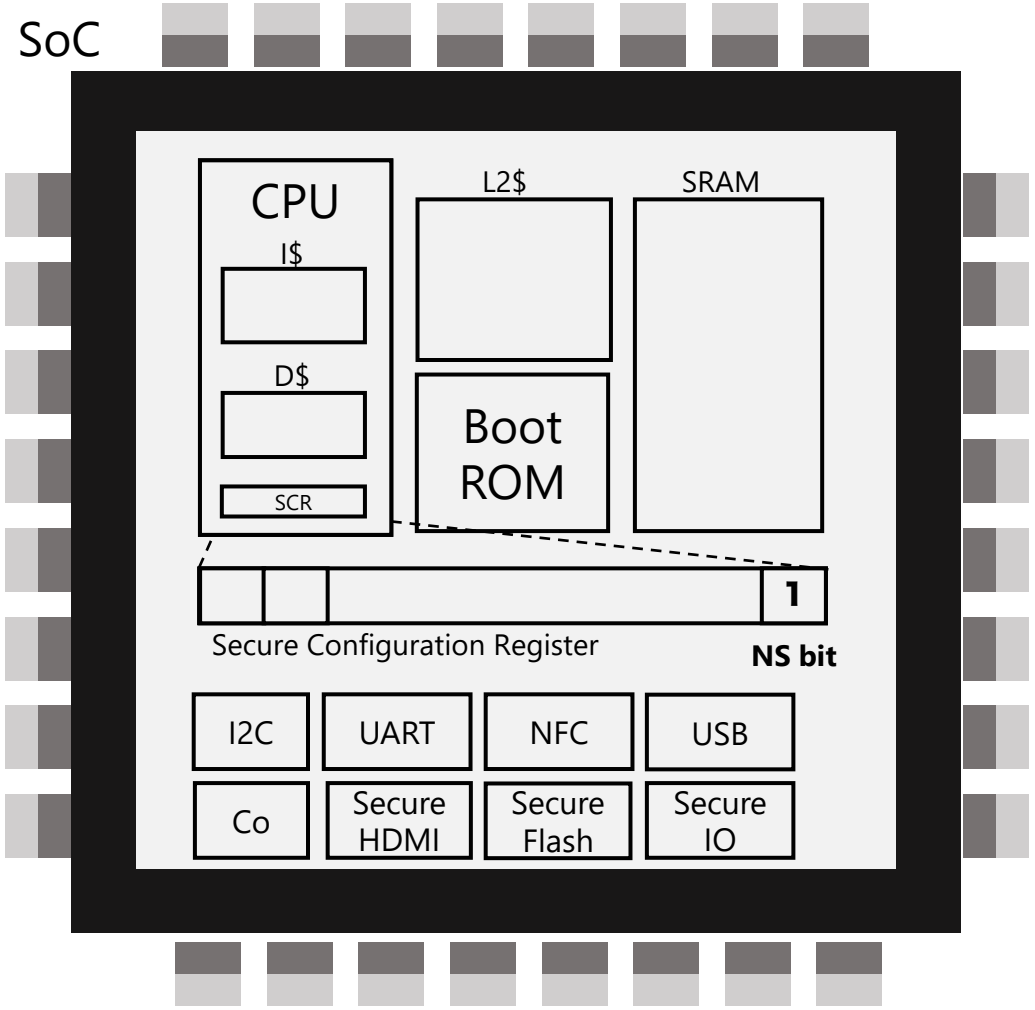
ARMv6 (2001-)

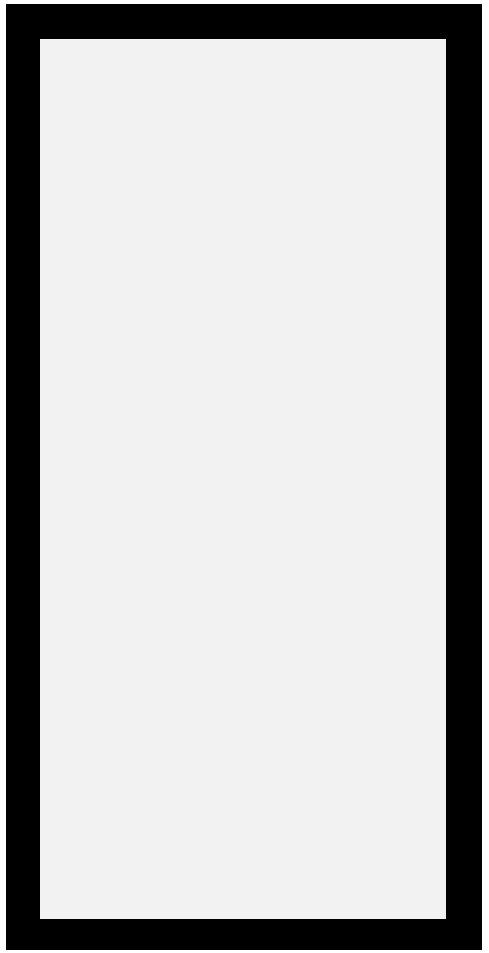


Non-Secure/Secureとアクセス制御

https://www.timesys.com/security/trusted-software-development-op-tee/ 掲載の図を参考に再作成

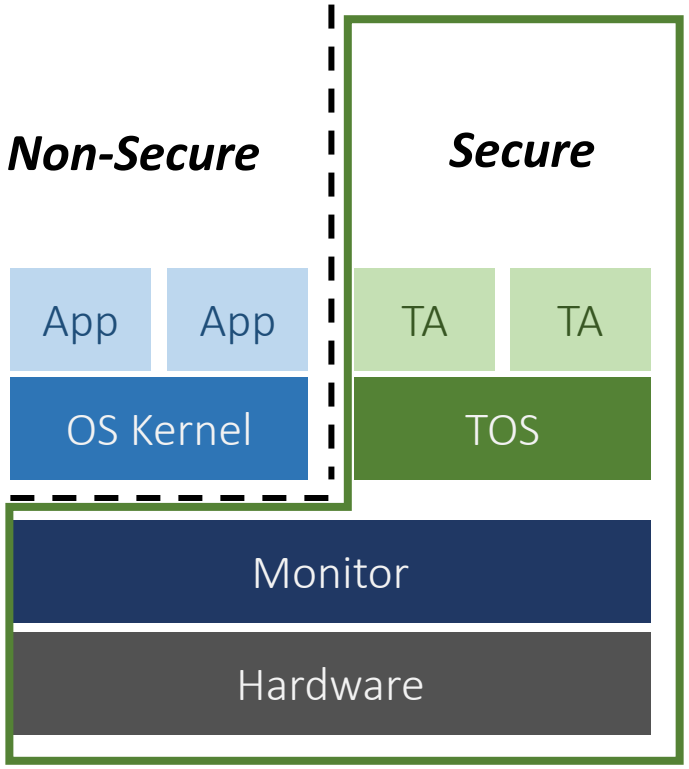


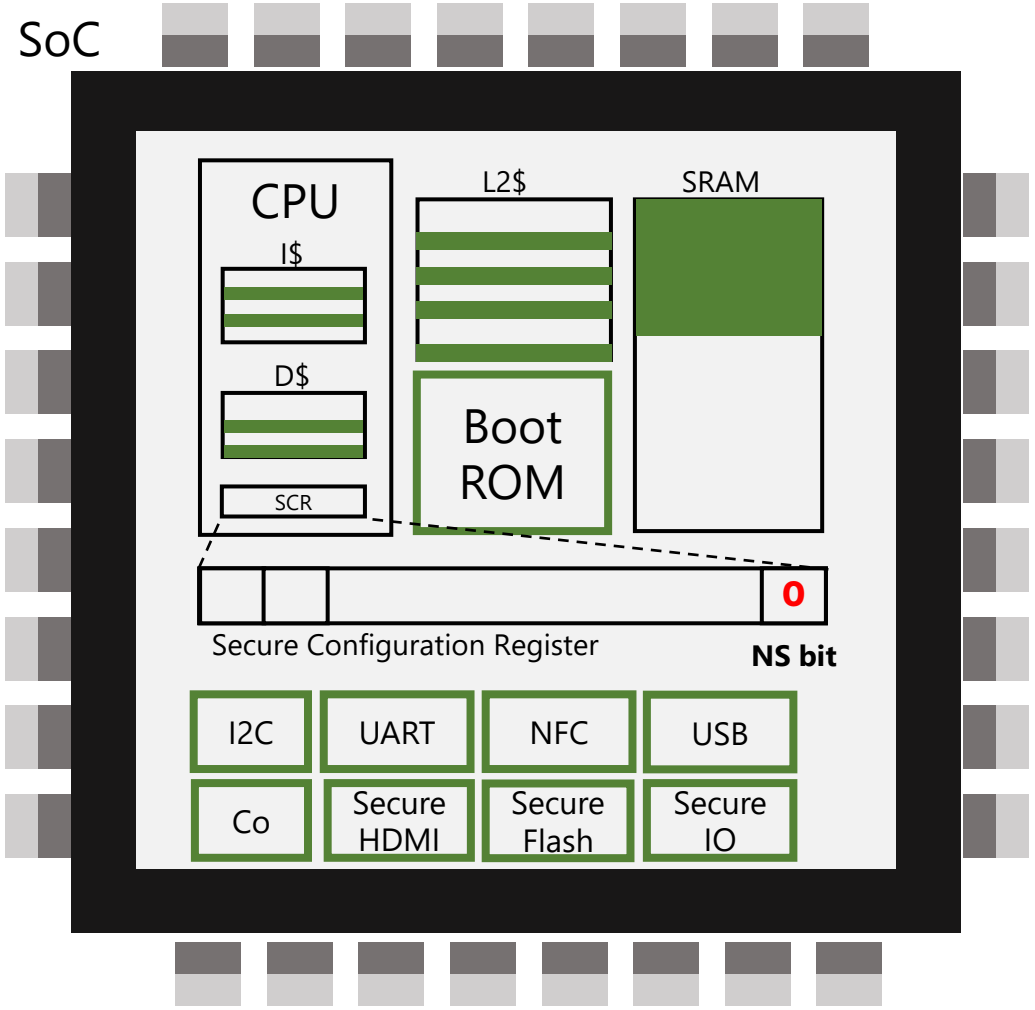


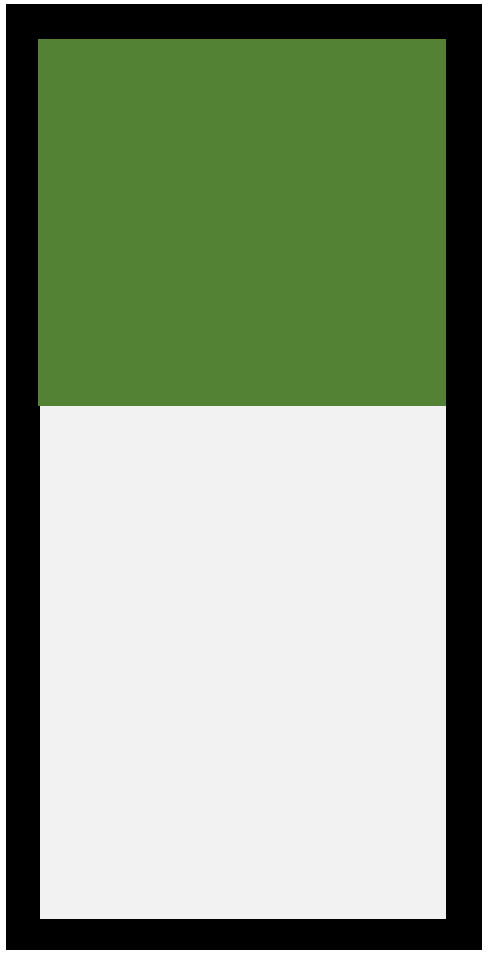


Secureの使用例(図は簡略化しています)

https://www.timesys.com/security/trusted-software-development-op-tee/ 掲載の図を参考に再作成

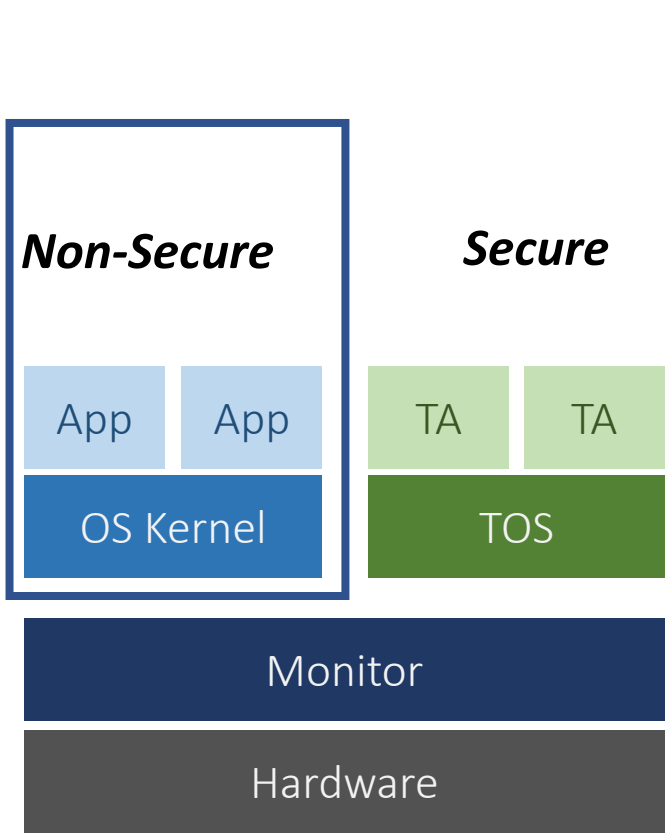


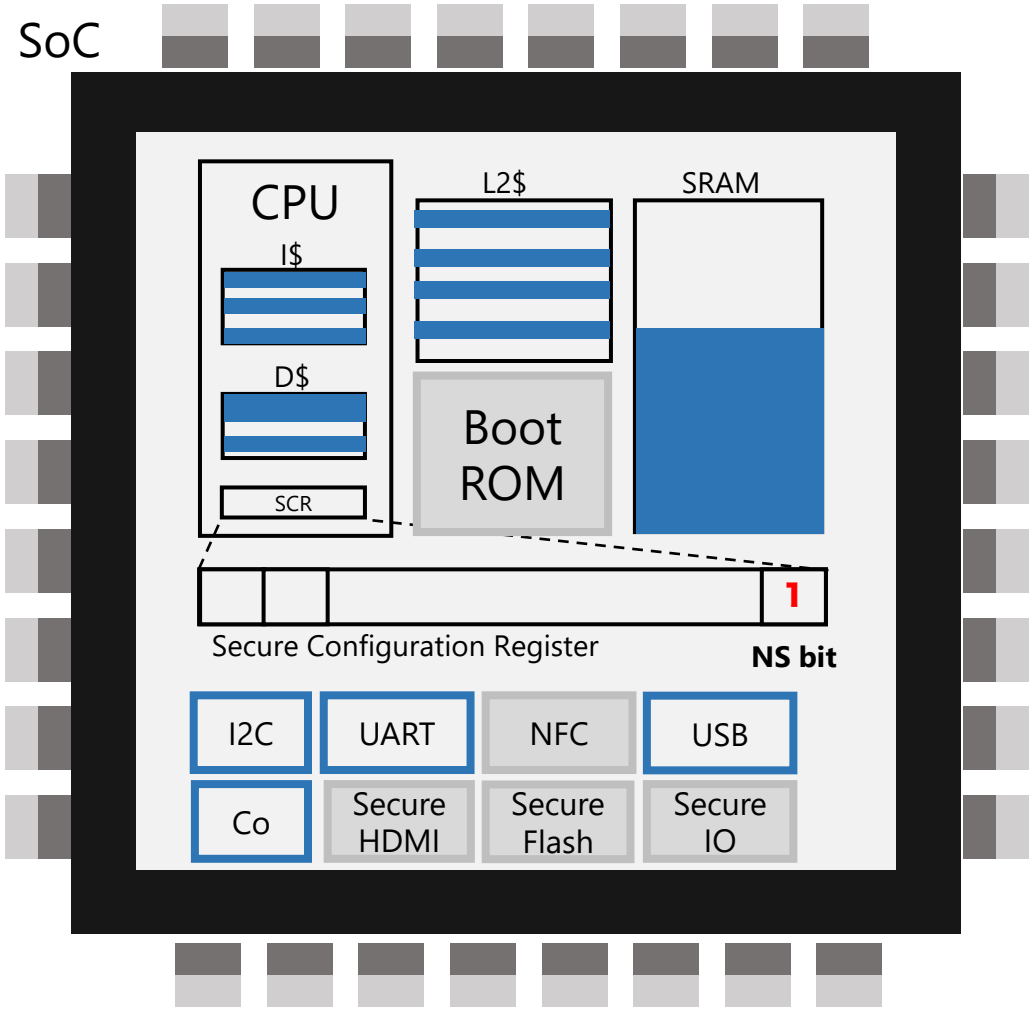


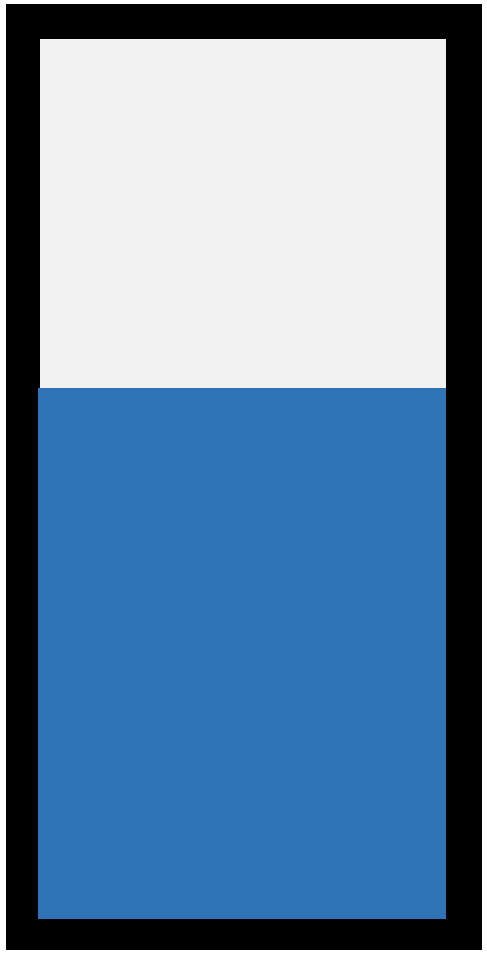


Non-Secureの使用例(図は簡略化しています)

https://www.timesys.com/security/trusted-software-development-op-tee/ 掲載の図を参考に再作成







アクセスをコントロールする回路

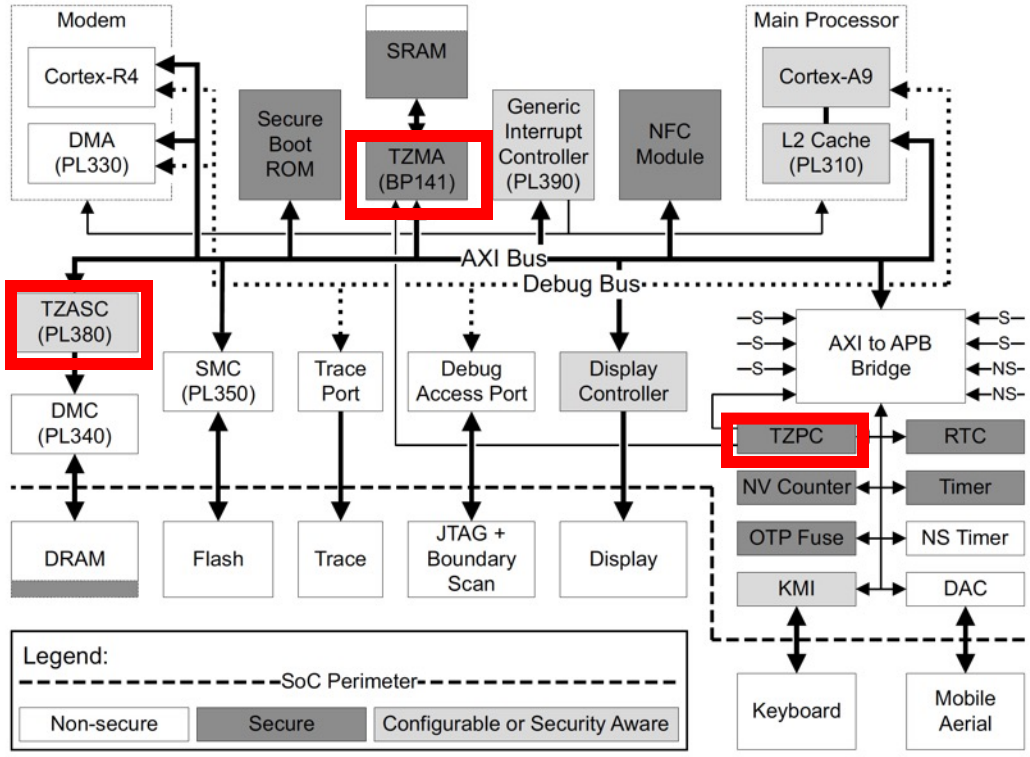


Figure 6-1 : The Gadget2008 SoC design

TZMA

The TrustZone Memory Adapter for on-SoC (SRAM)

TZASC

The TrustZone Address Space Controller for off-SoC (DRAM)

TZPC

The TrustZone Protection Controller for controling security signals on a SoC

ARMv7時代のマニュアル https://developer.arm.com/documentation/PRD29-GENC-009492/c/TrustZone-System-Design
Park, Heejin, and Felix Xiaozhu Lin. "Safe and Practical GPU Computation in TrustZone." Proceedings of the Eighteenth European Conference on Computer Systems. 2023.
Lentz, Matthew, et al. "Secloak: Arm trustzone-based mobile peripheral control." Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. 2018.

メモリ利用範囲の設定(OP-TEE)

optee_os/arch/arm/***/
platform_config.h

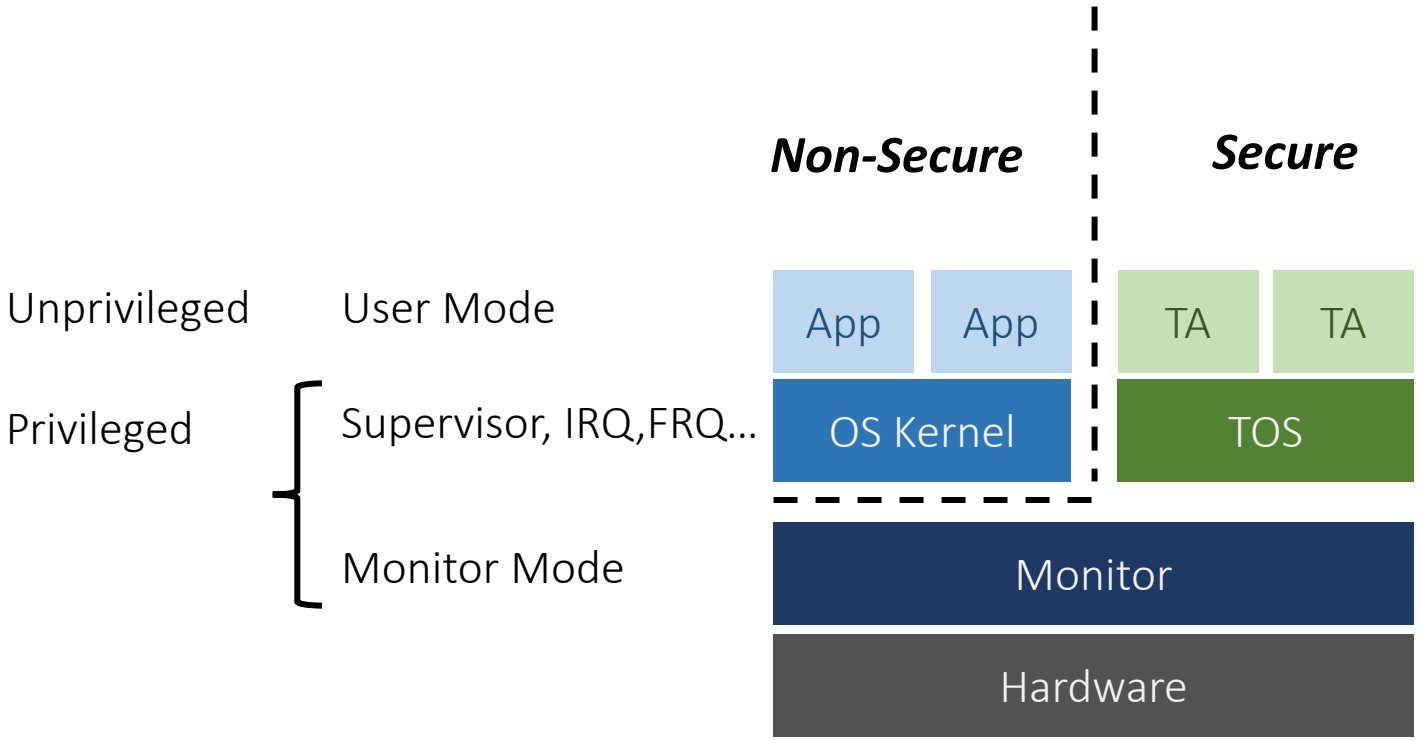
```
#ifndef PLATFORM_CONFIG_H
#define PLATFORM_CONFIG_H

/* Make stacks aligned to data cache line length */
#define STACK_ALIGNMENT             64

/* Optional: when used with CFG_WITH_PAGER, defines the device SRAM */
#define TZSRAM_BASE         0x3F000000
#define TZSRAM_SIZE         (200 * 1024)

/* Mandatory main secure RAM usually DDR */
#define TZDRAM_BASE         0x60000000
#define TZDRAM_SIZE         (32 * 1024 * 1024)

/* Mandatory TEE RAM location and core load address */
#define TEE_RAM_START               TZDRAM_BASE
#define TEE_RAM_PH_SIZE             TEE_RAM_VA_SIZE
...
```

https://optee.readthedocs.io/en/latest/architecture/core.html
https://github.com/apache/incubator-teaclave-trustzone-sdk/blob/master/docs/expanding-ta-secure-memory-on-qemuv8.md

ARMv6, ARMv7, ARMv8, ARMv9を駆け足で紹介

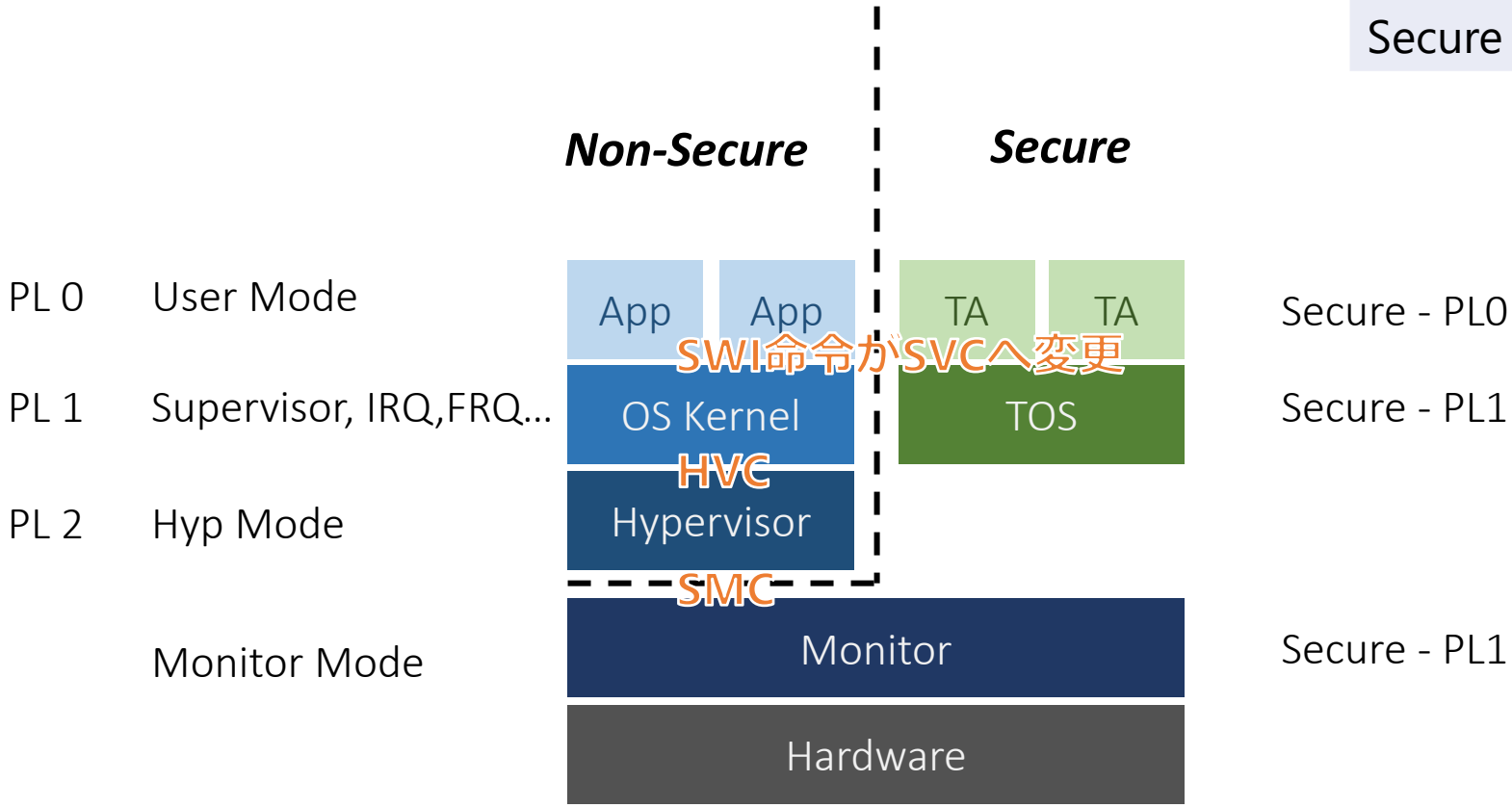
ARM v6 (2001-)

Memory PAS	Non Secure	Secure
Non-Secure	○	○
Secure	×	○



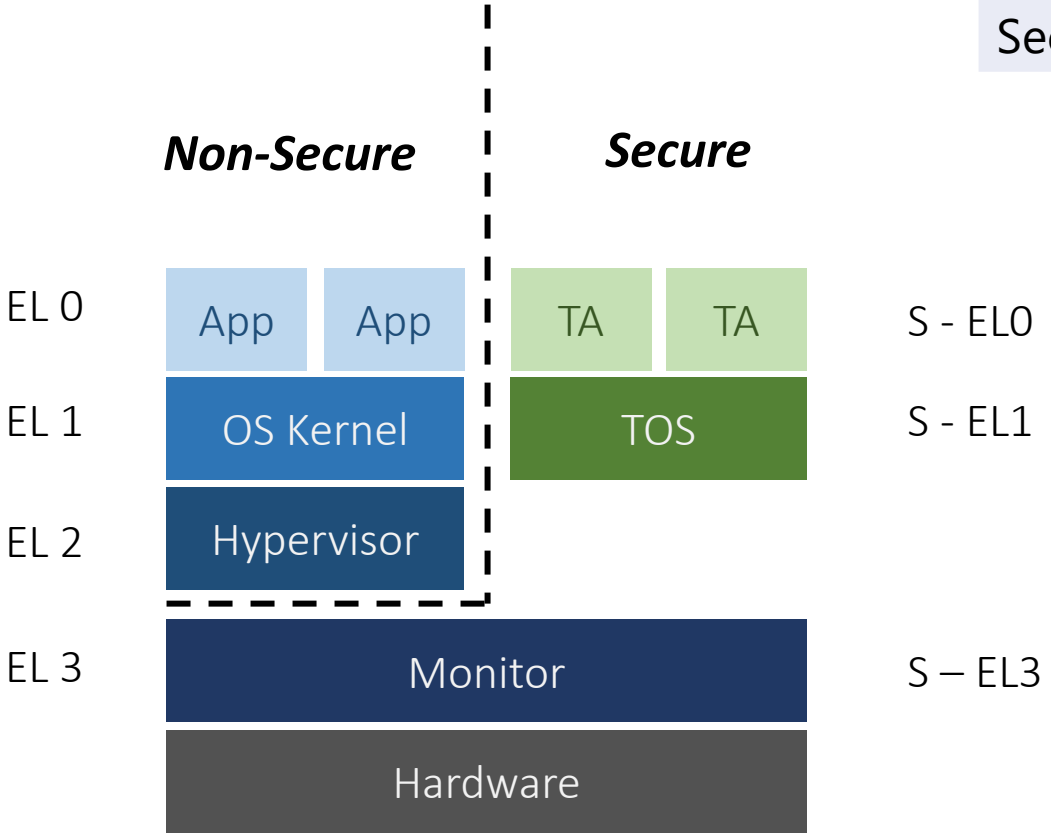
ARM v7 (2004-)

Memory PAS	Non Secure	Secure
Non-Secure	○	○
Secure	×	○



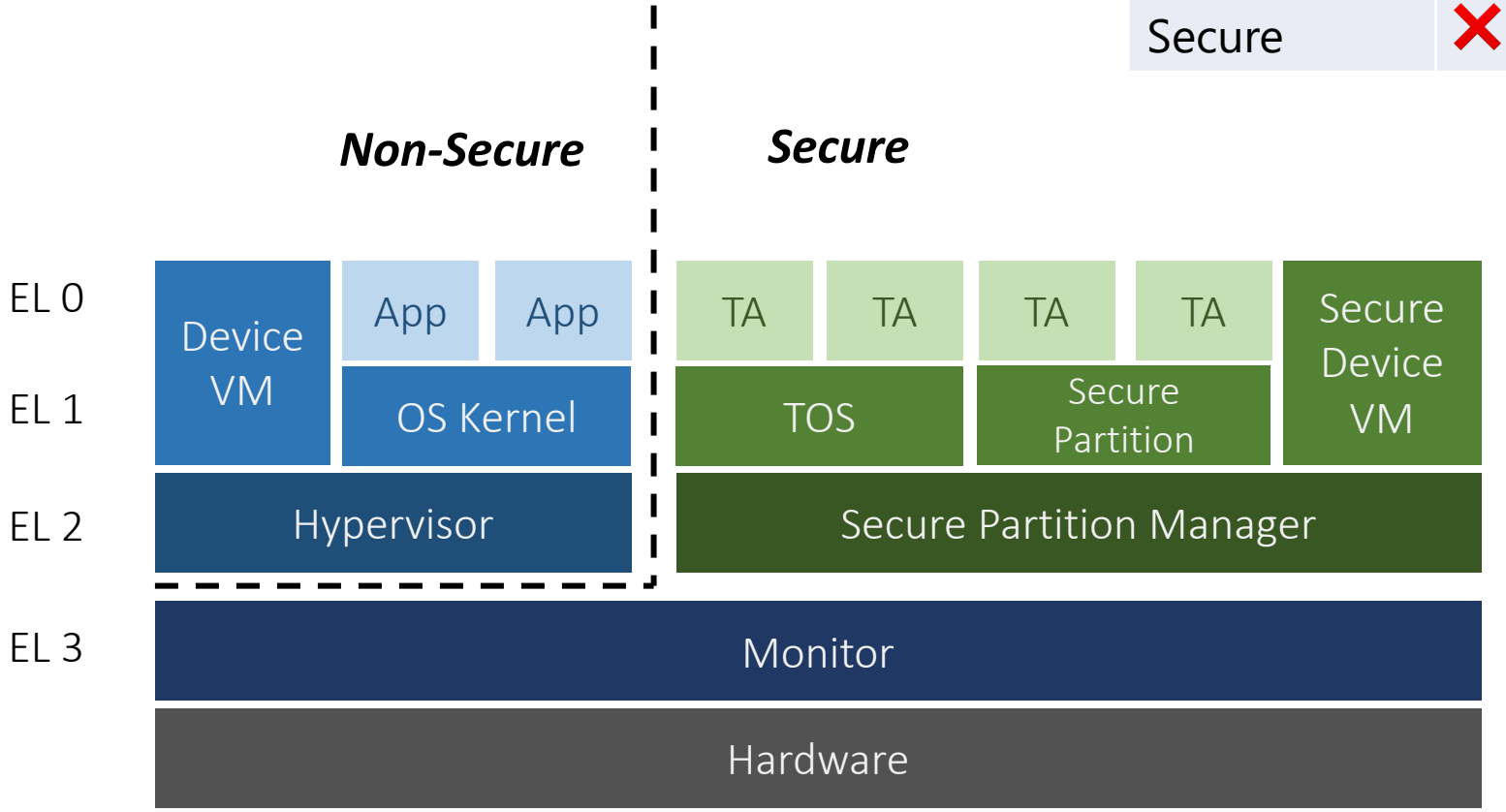
ARM v8-A(2011-)

Memory PAS	Non Secure	Secure
Non-Secure	○	○
Secure	×	○



ARMv8.4 (2014)

Memory PAS	Non Secure	Secure
Non-Secure	○	○
Secure	×	○

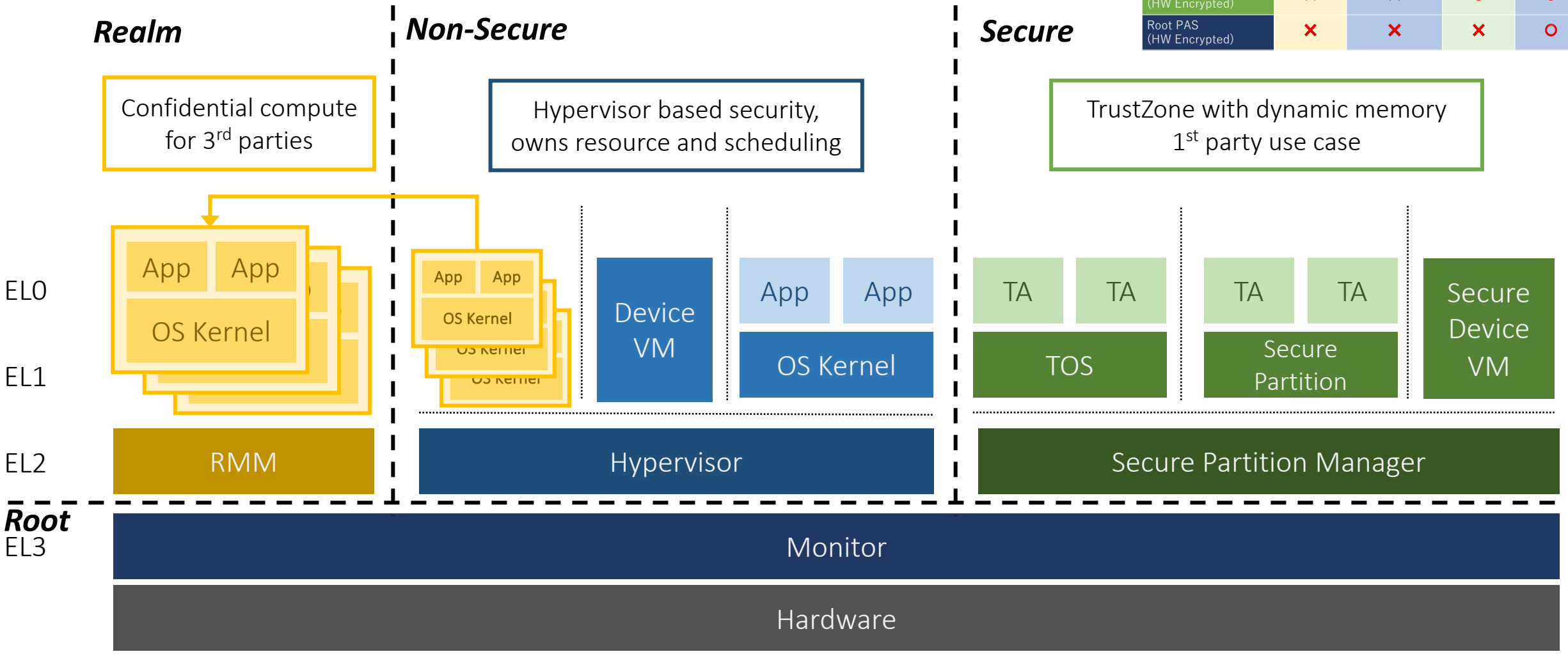


Secure Partition Managerのオープンソースソフトウェアプロジェクト「Hafnium」の開発が進められている。
https://www.trustedfirmware.org/projects/hafnium

Realm Management Monitorのオープンソースソフトウェア
プロジェクト「TF-RMM」の開発が進められている。
https://www.trustedfirmware.org/projects/tf-rmm

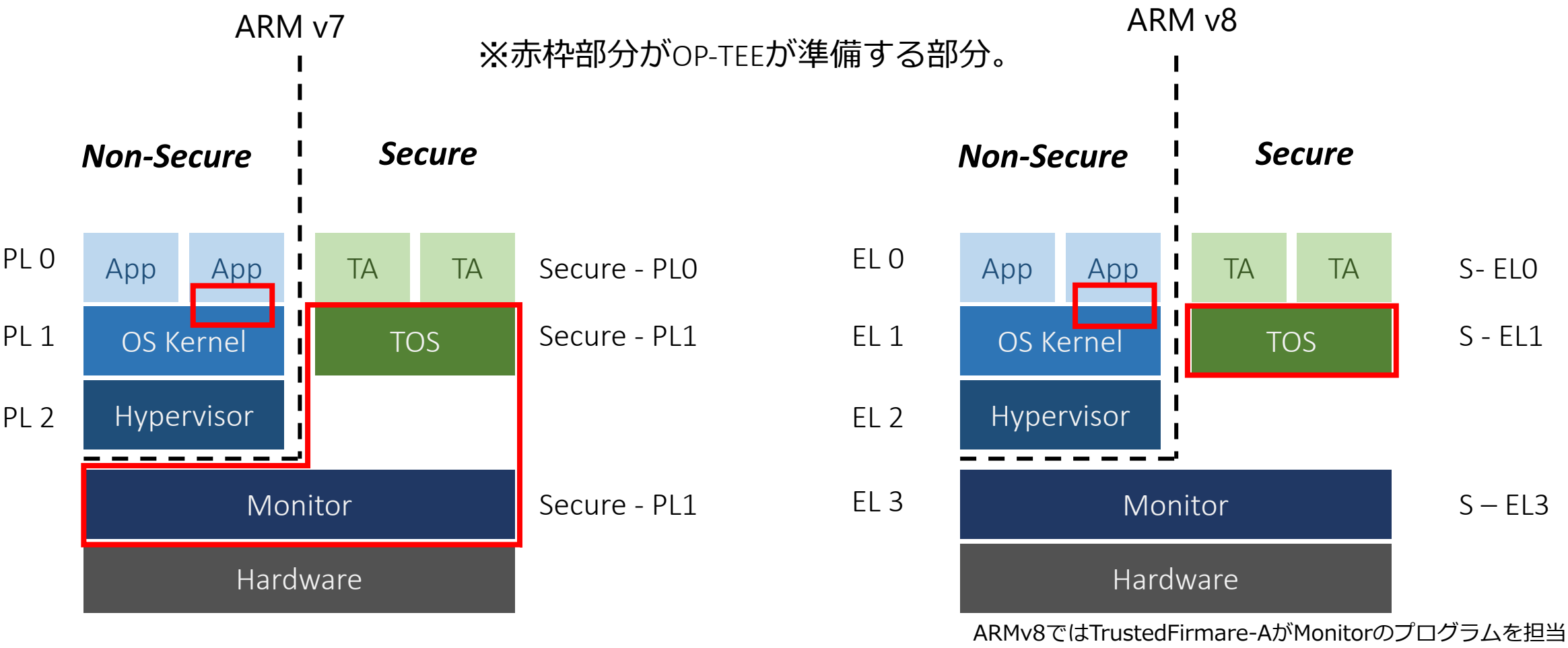
ARMv9.0 CCA (2021)まだ製品はない

Memory Physical Address Space (PAS)	Realm Program	Non-Secure Program	Secure Program	Root Program
Realm PAS (HW Encrypted)	○	×	×	○
Non-Secure PAS	○	○	○	○
Secure PAS (HW Encrypted)	×	×	○	○
Root PAS (HW Encrypted)	×	×	×	○



参考　https://trustedfirmware-a.readthedocs.io/en/latest/components/realm-management-extension.html

OP-TEEにはARMv7向けとARMv8向けがある



ARMv8ではTrustedFirmare-AがMonitorのプログラムを担当。

"Armv7-A platforms should use the OP-TEE OS secure monitor. Armv8-A platforms are likely to rely on an Trusted Firmware A."
https://optee.readthedocs.io/en/latest/architecture/core.html#core-exception-vectors

技術仕様の標準化とGlobalPlatform

モバイルセキュリティ技術の標準化（TCG）

- **TCG (2003-)**
  - TPMチップで有名
  - Intel, Microsoft, AMD, HP, IBM, Sun Microsystemsなど、コンピューターを製造しているメーカーによって作られた団体
  - Trusted Computingを実現するためのハードウェアやソフトウェアの検討
  - Trusted Computing Platform Alliance (1999-2003) が元の団体
- **Nokiaのモチベーション**
  - 仕様のメンテナンスをNokiaが負担する必要がなくなる
  - チップ技術が標準化されれば、チップベンダー間で競争が起きる
- **Trusted Computing Group（TCG）への提案**
  - Mobile Phone Work Groupを主催 (2005)
  - Mobile Trusted Platform Module (Mobile TPM, MTM)を発表 (2007)

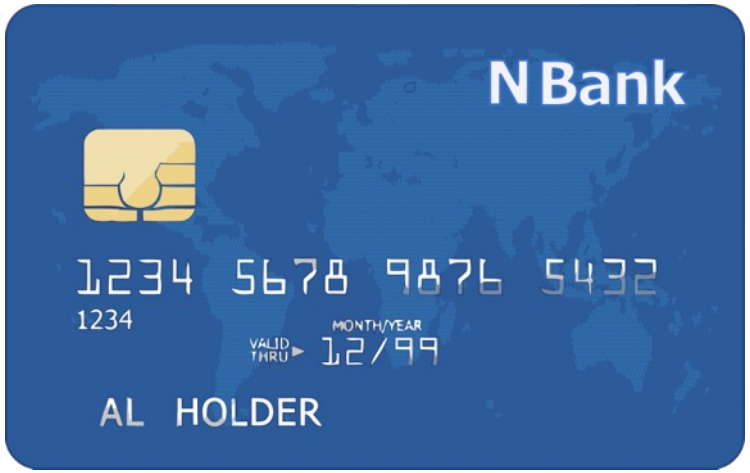
参考文献：https://blog.ssg.aalto.fi/2019/06/historical-insight-into-development-of.html

モバイルセキュリティ技術の標準化（OMTP）

- **OMTP (2004-)**
  - Open Mobile Terminal Platform
- **概要**
  - モバイルセキュリティ技術の標準化（TCG）
  - 今後の携帯端末に搭載される端末プラットフォーム技術の検討を行うコンソーシアム
- **活動内容**
  - オープンな携帯電話プラットフォームを策定
  - 端末メーカーやソフトウェアベンダーに採用を提案
  - 標準化活動のロードマップも制定
  - 各種標準化団体などへの標準化の働きかけ
- **設立メンバー**
  - Telecom Italia Mobile, mmO2, Vodafone, Telefo'nica Mo'viles, SMART Communications, Orange, T-Mobilem, NTTドコモ
- **Advanced Trusted Environment(TR0: 2006, TR1: 2008)**
  - モバイル環境に必要なセキュリティ基盤に関するドキュメント
  - **初めてTrusted Execution Enviroment (TEE)という言葉を使った**

https://en.wikipedia.org/wiki/Open_Mobile_Terminal_Platform
http://www.omtp.org/OMTP_Trusted_Environment_OMTP_TR0_v1_2.pdf
http://www.omtp.org/OMTP_Advanced_Trusted_Environment_OMTP_TR1_v1_1.pdf

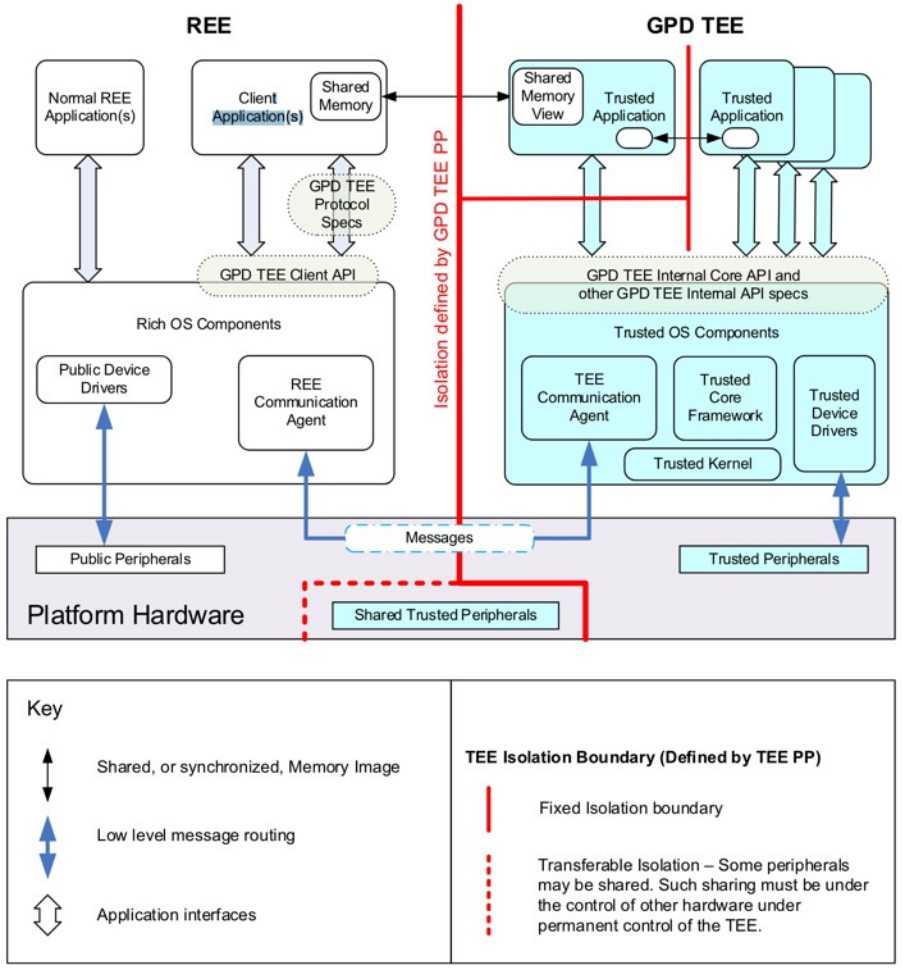
モバイルセキュリティ技術の標準化（TCG）



- **GlobalPlatform (1999-)**
- **設立経緯**
  - 1996
    - Schlmberger：Java Card発表
  - 1998
    - Visa：Visa Java Open Platform（仕様）
  - 1999
    - Visa, NTT 他15社：
      Global Platform設立（仕様・組織名）
- **仕様・組織のモチベーション**
  - Java Card上のアプリの移植性や、Java Cardと読み取り装置と通信など、相互運用に重要な仕様の取り決め。
  - 現在では、仕様に準拠しているかの認証も行っている。

https://xtech.nikkei.com/dm/article/NEWS/20070719/136272/
https://globalplatform.org/latest-news/globalplatform-attracts-new-members/

Global Platformで提案している仕様

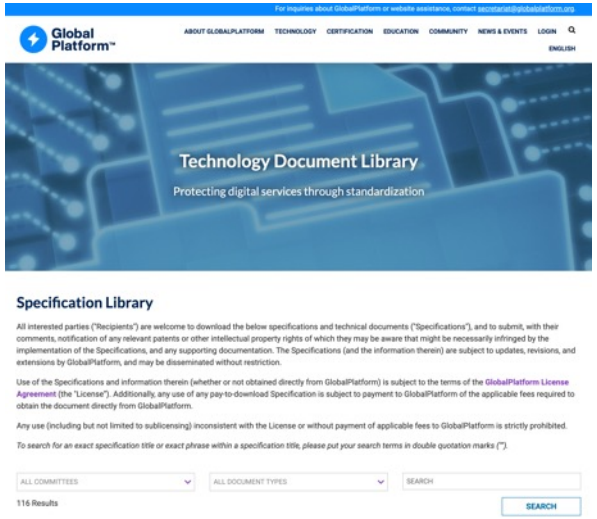


- **GPD TEE System Architecture**
  - TEEを実現するアーキテクチャ全体の仕様
- **GPD TEE Client API**
  - REE側アプリで使うAPI仕様
- **GPD TEE Internal Core API**
  - TEE側のアプリ（Trusted Application）で使うAPI仕様
- **その他**

GlobalPlatform Technology “TEE System Architecture Version 1.2” (2018)から引用

Global Platformの仕様ドキュメント

- **TEE関連のドキュメントを検索すると29件ヒット（2024/05/07）**
  - TEE System Architecture v1.3 (2022)
  - TEE Client API Specification v1.0 (2010)
  - TEE Internal Core API Specification v1.3.1 (2021)
  - TEE Sockets API Specification v1.0.1, 1.0.3 & 1.1 (2022)
  - …



https://globalplatform.org/specs-library/

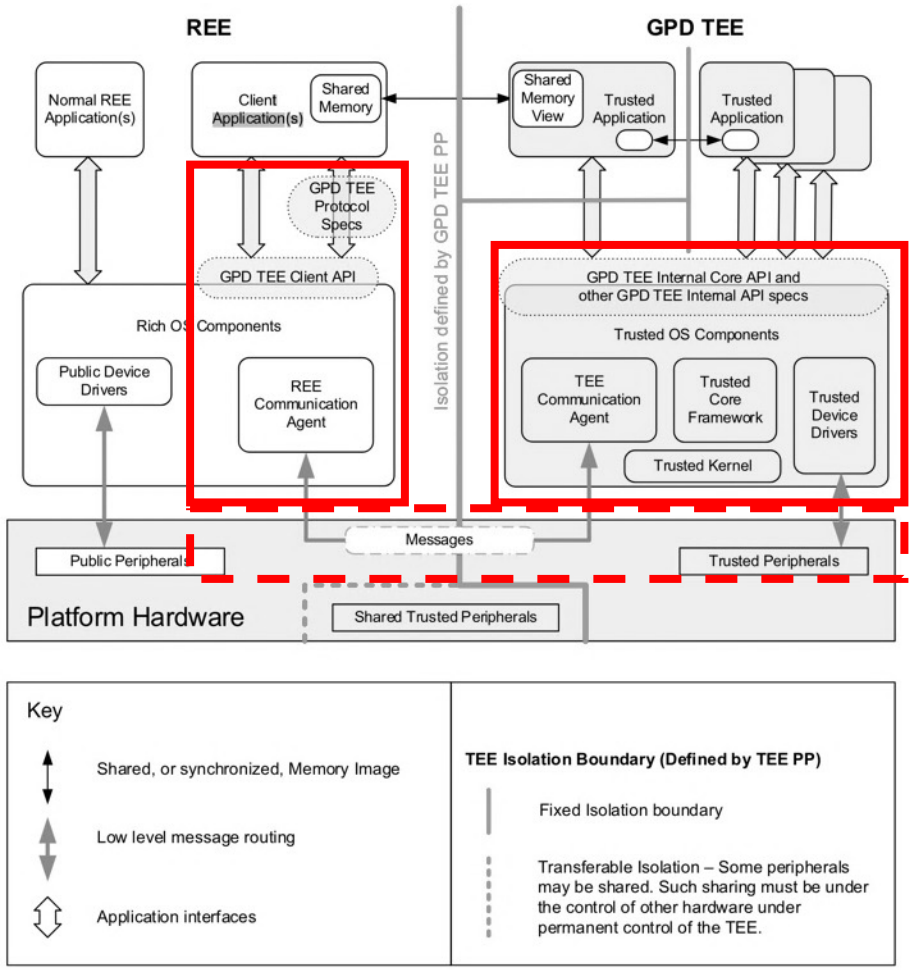
OP-TEE概要

OP-TEE (Open Portable TEE)誕生の経緯

- **???-2013**
  - 各社独自TEEシステムソフトウェアを開発
  - ST-Ericson（現STMicroelectronics）も独自TEEシステムソフトウェアを開発
- **2013**
  - ST-EricsonがGlobalPlatformからコンプライアンス認定取得
- **2013 後半**
  - LinaroがセキュリティWGを形成
    - オープンソースのTEEシステムソフトウェアを開発しようとして、さまざまなベンダーと交渉。ST-Ericsonが協力することになった。
  - ※Linaro：英国の非営利組織。2010年からArm向けのオープンソースソフトウェア開発を推進している。
- **2014/06/12**
  - オープンソースプロジェクト**OP-TEE**としてGitHubで公開
- **2015年**
  - OP-TEEの所有権がSTMicroelectronicsからLinaroに移管される。
- **2019年9月**
  - OP-TEEの所有権がLinaroからTrustedFirmware.orgプロジェクトに移管される。

https://optee.readthedocs.io/en/latest/general/about.html#history

OP-TEEが提供するもの



- **GPD TEE Client API**
  - REE側アプリで使うAPI
- **GPD TEE Internal Core API**
  - TEE側のアプリ（Trusted Application）で使うAPI
- **Trusted OS**
  - OP-TEE OS
- **Secure Monitor**
  - ARMv7ではOP-TEEが提供
  - ARMv8ではTrusted Firmware-Aが提供
- **その他**
  - Linux用ドライバ(Linux v4.12で本家へマージされた)
  - REE Commincation Agent
  - SDK
  - テスト
  - サンプルコード
  - 等々

GlobalPlatform Technology “TEE System Architecture Version 1.2” (2018)から引用

前半まとめ

- **ARM**
  - ARMはv1, v2...のように進化している
  - ARMv6のころに、モバイル系のセキュリティ対策としてTrustZoneが生まれた
  - ARMv7とARMv8で大きく仕様が変わった
- **TrustZone**
  - 1つのCPUをあたかも2つのCPUのように独立して振る舞わせる技術
  - SecureWorldとNormal Worldの２つがある
  - ハードウェアだけでなくシステムソフトウェアも重要な技術
- **GlobalPlatformと標準化**
  - TrustZoneの開発陣営は仕様を標準化させたかった
  - TCG、OMTP、GlobalPlatformからそれぞれモバイルデバイスのセキュリティ仕様について提案があった
  - OP-TEEはGlobalPlatformの仕様に準拠している
- **OP-TEE (Open Portable TEE)**
  - ST-Ericsonが独自のTEEシステムソフトウェアを作っていた
  - そのシステムソフトウェアがGlobalPlatformの仕様準拠の認証が得られた
  - 同じ時期に、LinaroがTEEシステムソフトウェアのオープンソース化を目指しており、STーEricsonが協力し、2014年からオープンソースとなった
  - ARMv7向けとARMv8向けで異なるアーキテクチャになっている

ワークショップ

実際に手元のPCで中身を見ながら聞いて下さい

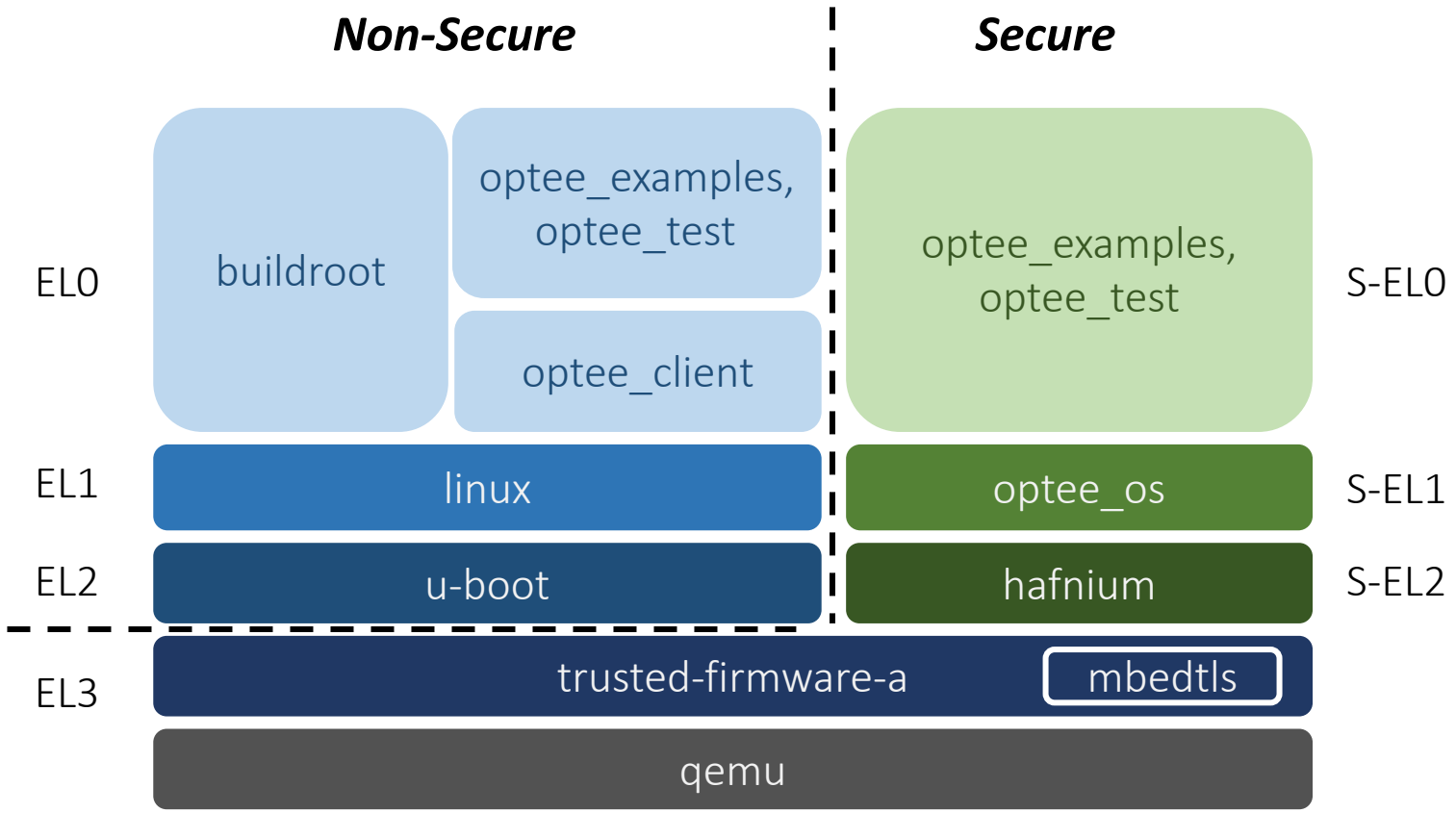
ディレクトリ構成（Ubuntu開発環境全体）

```
optee/
├── build/
├── buildroot/
├── hafnium/
├── linux/
├── mbedtls/
├── optee_client/
├── optee_examples/
├── optee_os/
├── optee_rust/
├── optee_test/
├── out/
├── out-aarch64-sdk/
├── out-br/
├── qemu/
├── toolchains/
├── trusted-firmware-a/
└── u-boot/
```

全ディレクトリの内容を説明しました。

赤字のディレクトリは、OP-TEEの仕組みを知るときによく見るディレクトリ

緑色のディレクトリは、Trusted Applicationを開発するときによく参考にするディレクトリ

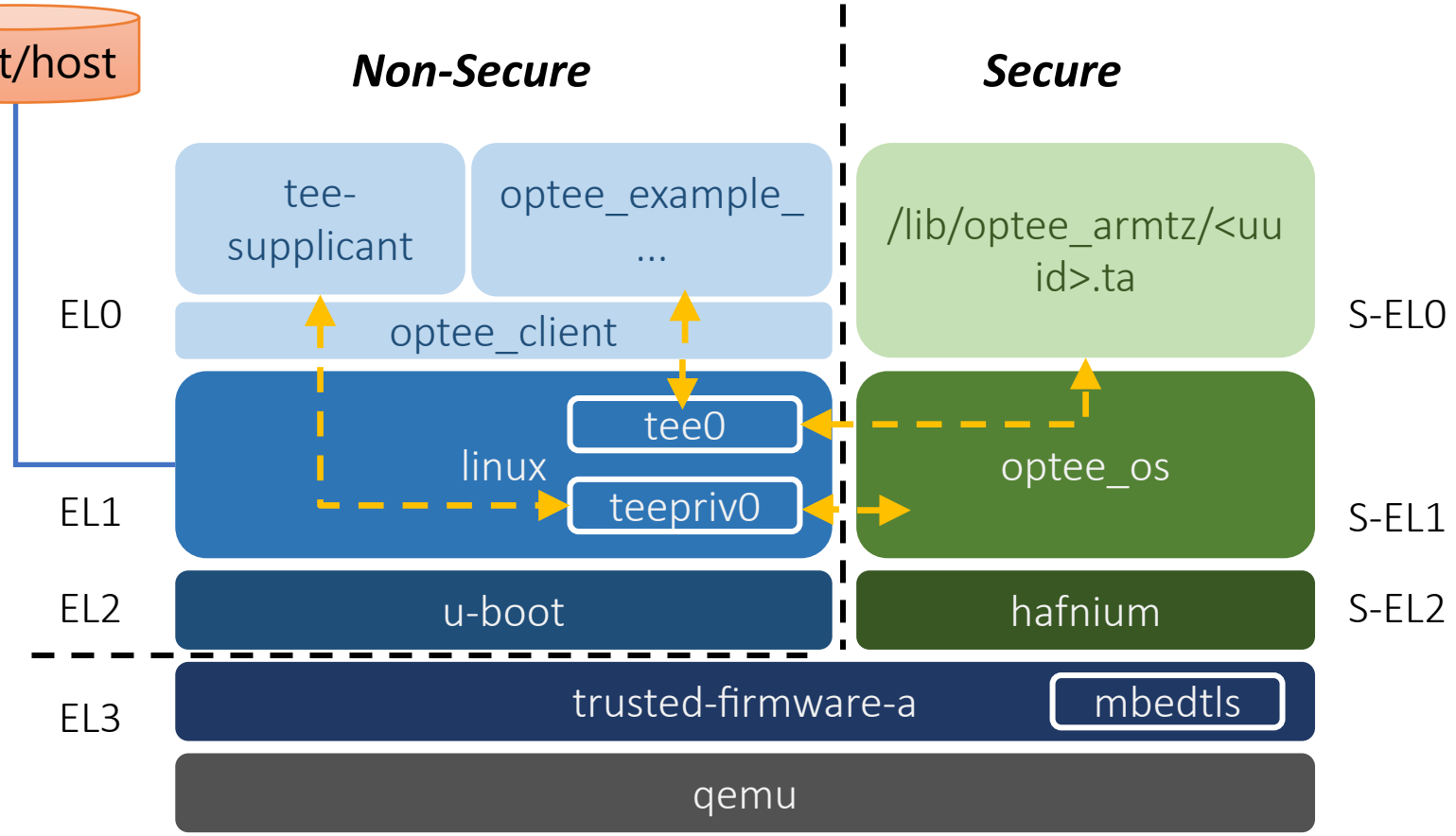


実際に手元のPCで中身を見ながら聞いて下さい

ディレクトリ構成（QEMU上Linux）

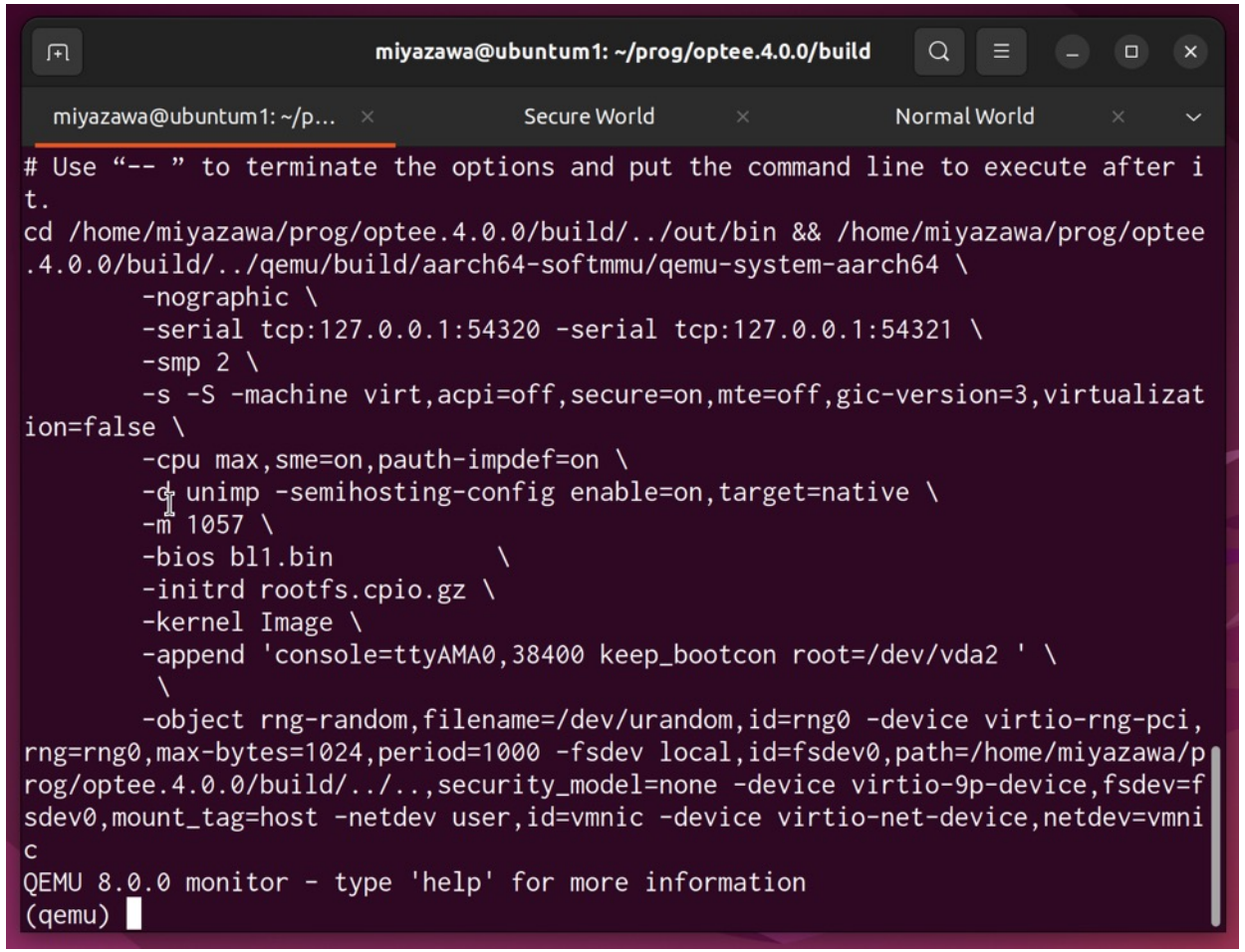
全ディレクトリの内容を説明しました。
赤字のディレクトリは、TEEの開発で知っておくべきディレクトリ

```
/
├── bin/
├── data/tee/
├── dev/tee0
├── dev/teepriv0
├── etc/
├── lib/optee_armtz/<uuid>.ta
├── mnt/host/
├── proc/
├── root/
├── run/
├── sbin/
├── sys/
├── usr/bin/
├── usr/sbin/tee-supplicant
├── .../
└── var/
```



実際に手元のPCで動かしながら聞いて下さい

実行方法 （GUI）



実際に手元のPCで動かしながら聞いて下さい

実行方法（ネットワーク越しCUI）

(3) QEMU

```
$ cd ~/optee/build
$ make run-only
QEMU_VIRTFS_ENABLE=y
QEMU_VIRTFS_HOST_DIR=$(pwd
)/../..
.
.
```

(1) REE (ネットワーク越し）

```
$ cd ~/optee/build
$ ./soc_term.py 54320
...
Welcome to Buildroot, type root or test to login
buildroot login:
```

(2) TEE (ネットワーク越し）

```
$ cd ~/optee/build
$ ./soc_term.py 54321
...
D/TC:? 0 tee_ta_close_session:468 csess 0xe19c860 id 2
D/TC:? 0 tee_ta_close_session:487 Destroy session
```

実際に手元のPCで動かしながら聞いて下さい

QEMUからホストのディレクトリをマウント

(1) QEMU

```
$ cd ~/optee/build
$ make run-only QEMU_VIRTFS_ENABLE=y QEMU_VIRTFS_HOST_DIR=$(pwd)/../..
.
.
(qemu) c
```

(2) REE

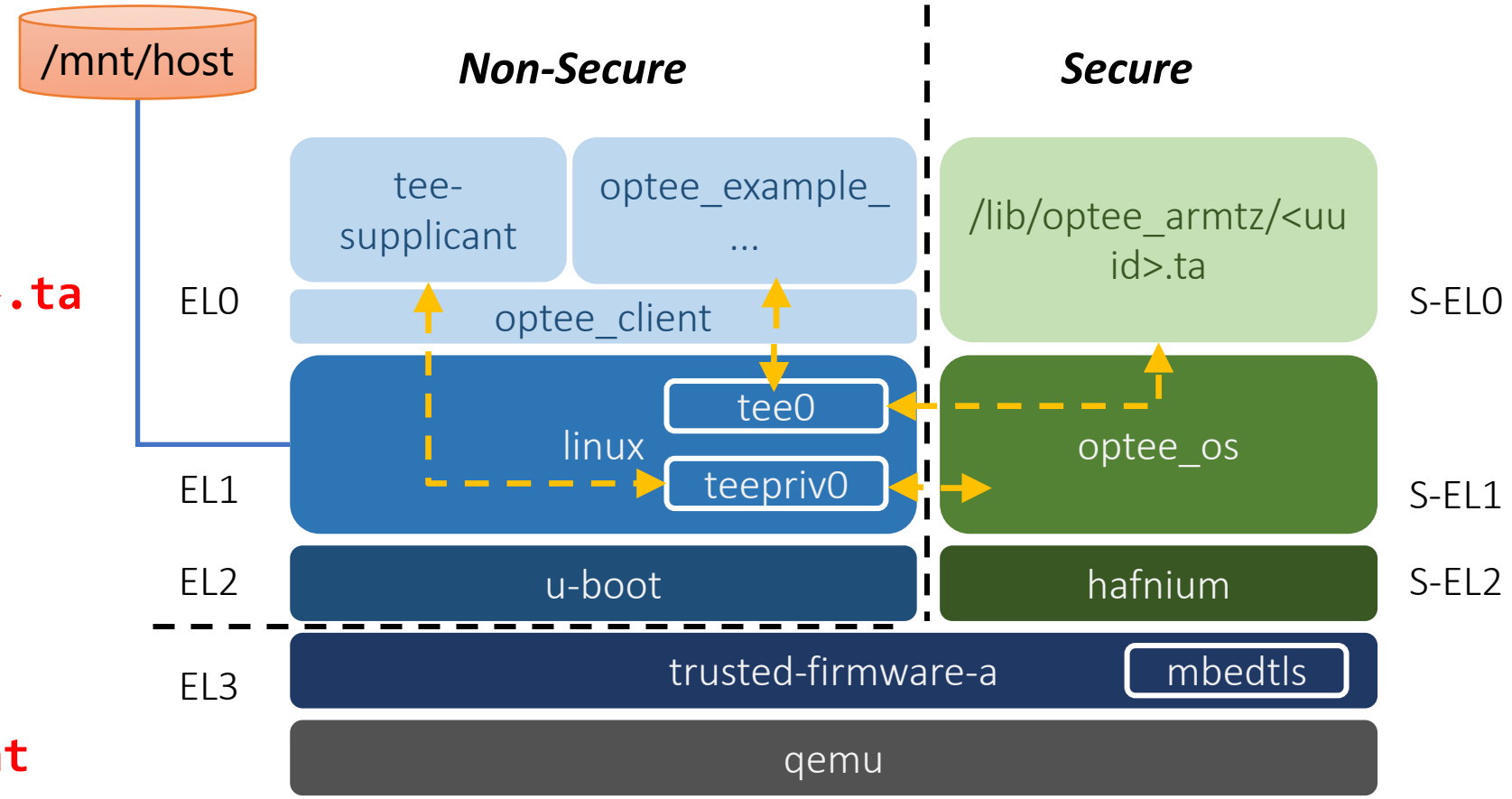
```
# mount -t 9p -o trans=virtio host /mnt/host
# cd /mnt/host
```

実際に手元のPCで中身を見ながら聞いて下さい

再掲

ディレクトリ構成（QEMU上Linux）

```
/
├── bin/
├── data/tee/
├── dev/tee0
├── dev/teepriv0
├── etc/
├── lib/optee_armtz/<uuid>.ta
├── mnt/host/
├── proc/
├── root/
├── run/
├── sbin/
├── sys/
├── usr/bin/
├── usr/sbin/tee-supplicant
├── .../
└── var/
```

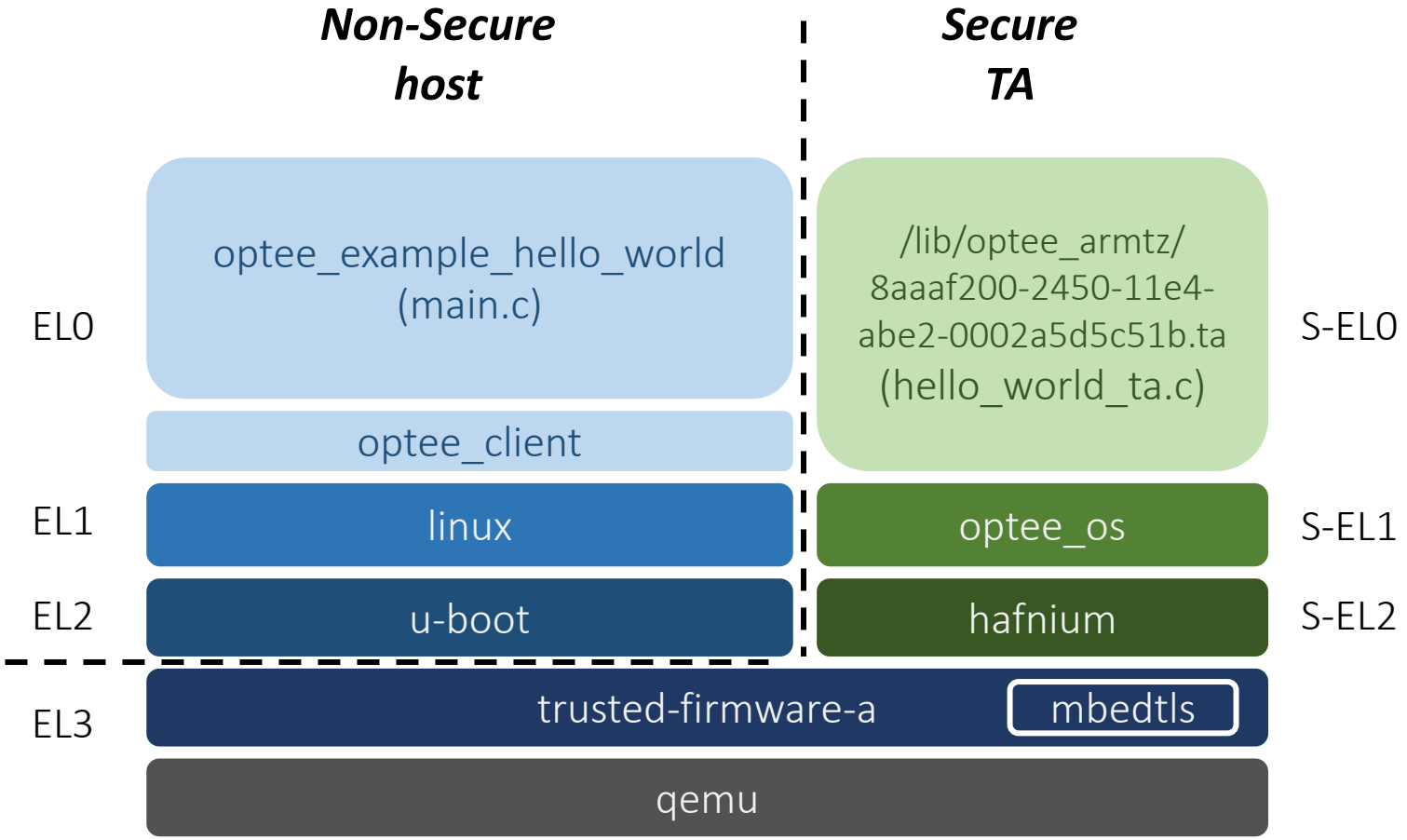


実際に手元のPCで中身を見ながら聞いて下さい

ディレクトリ構成（単体アプリ開発環境）

main.cと赤字のファイルのみ内容を説明しました。
赤字のファイルは、Trusted Applicationを開発する上で書き方の作法が決まっているファイル

```
optee_examples/hello_world/
├── Android.mk
├── CMakeLists.txt
├── host/
│   ├── main.c
│   └── Makefile
├── Makefile
└── ta/
    ├── Android.mk
    ├── hello_world_ta.c
    ├── include/
    │   └── hello_world_ta.h
    ├── Makefile
    ├── sub.mk
    └── user_ta_header_defines.h
```



https://optee.readthedocs.io/en/latest/building/trusted_applications.html

ファイル内容（単体アプリ開発環境）

Makefile

```
BINARY=8aaaf200-2450-11e4-abe2-0002a5d5c51b
include $(TA_DEV_KIT_DIR)/mk/ta_dev_kit.mk
```

sub.mk

```
srcs-y += hello_world_ta.c
global-incdirs-y += include/
cflags-hello_world_ta.c-y += -Wno-strict-prototypes
cflags-remove-hello_world_ta.c-y += -Wno-strict-prototypes
libnames += foo
libdirs += path/to/libfoo/install/directory
libdeps += path/to/greatlib/libgreatlib.a
```

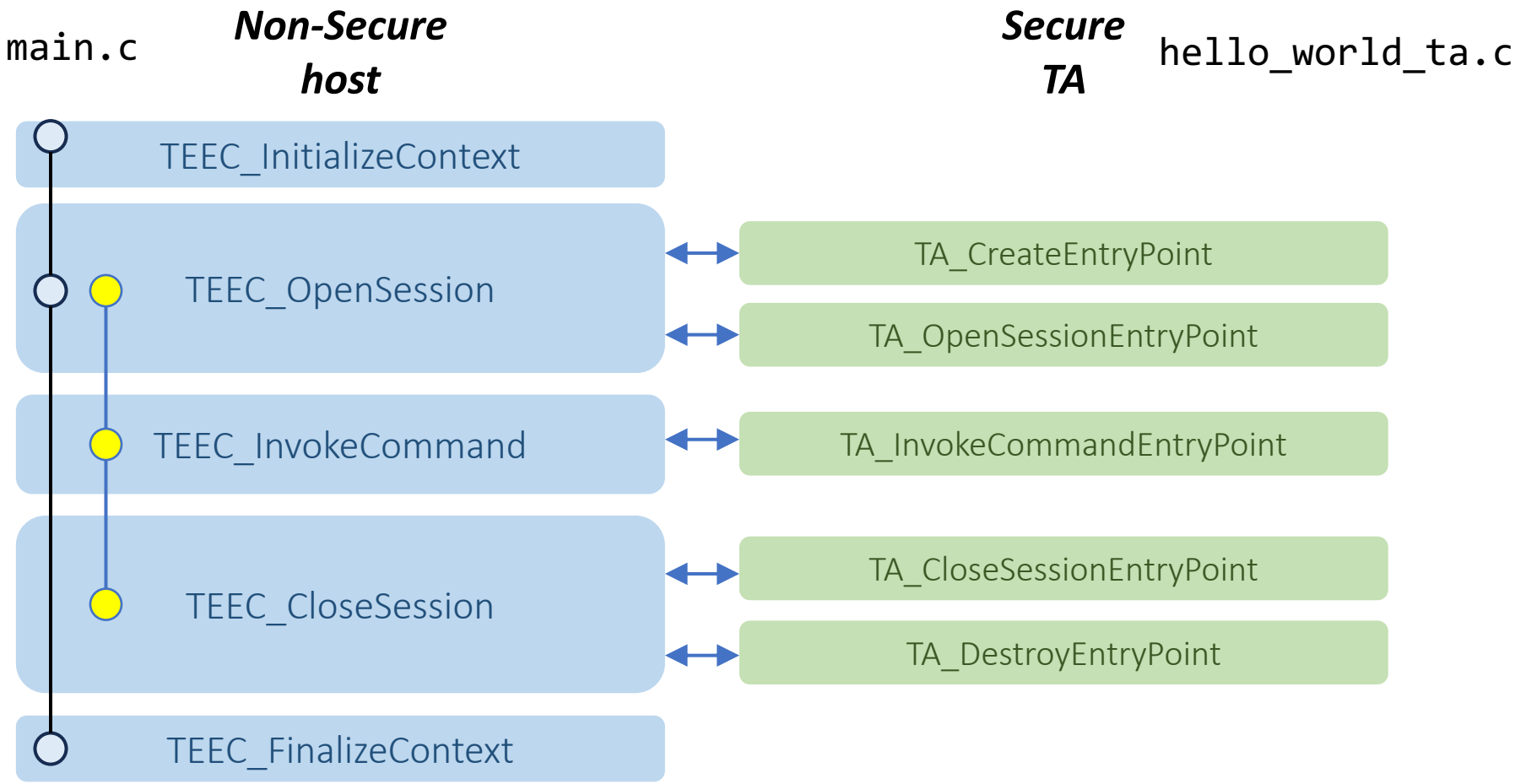
user_ta_header_defines.h

```
#define TA_UUID
    { 0x8aaaf200, 0x2450, 0x11e4, ¥
        { 0xab, 0xe2, 0x00, 0x02, 0xa5, 0xd5, 0xc5, 0x1b} }

#define TA_FLAGS                    (TA_FLAG_EXEC_DDR | ¥
                        TA_FLAG_SINGLE_INSTANCE | ¥
                        TA_FLAG_MULTI_SESSION)
#define TA_STACK_SIZE                       (2 * 1024)
#define TA_DATA_SIZE                        (32 * 1024)

#endif /* USER_TA_HEADER_DEFINES_H */
```

https://optee.readthedocs.io/en/latest/building/trusted_applications.html

実際に手元のPCで中身を見ながら聞いて下さい

TEEとREEの連携のフレームワーク



https://optee.readthedocs.io/en/latest/building/trusted_applications.html

実際に
optee_examples/hello_world
の中身を見ていこう！

画面を切り替えます

実際に手元のPCで動かしながら聞いて下さい

DEBUG: GDB (Normal World)

https://optee.readthedocs.io/en/latest/building/devices/qemu.html

(1) QEMU

```
$ cd ~/optee/build
$ make run GDBSERVER=y
.
.
(qemu) c
```

(2) REE (Linux)

```
# gdbserver :12345 xtest 4002
gdbserver: Unable to determine ...
gdbserver: Unable to determine ...
Process xtest created; pid = 138
Listening on port 12345
Remote debugging from host 10.0.2.2, port 52828
```

(3) GDB

```
$ cd ~/optee
$ out-br/host/bin/aarch64-linux-gdb -q
(gdb) set sysroot out-br/host/arm-buildroot-linux-gnueabihf/sysroot
(gdb) target remote :12345
(gdb) b main
(gdb) c
```

実際に手元のPCで動かしながら聞いて下さい

DEBUG: GDB (Secure World)

https://optee.readthedocs.io/en/latest/building/devices/qemu.html

(1) QEMU

```
$ cd ~/optee/build
$ make run CFG_CORE_ASLR=n
.
.
(qemu) c
```

(3) REE (Linux)

```
# xtest 4002
Test ID: 4002
Run test suite with level=0

TEE test application started over default TEE
instance
...
```

(2) GDB

```
$ cd ~/optee
$ out-br/host/bin/aarch64-linux-gdb -q
(gdb) target remote :1234
(gdb) symbol-file optee_os/out/arm/core/tee.elf
(gdb) b tee_entry_std
(gdb) c
```

実際に手元のPCで動かしながら聞いて下さい

DEBUG： Panic発生時のdumpメッセージ

TEE

```
E/TC:? 0
E/TC:? 0 TA panicked with code 0xffff0006
E/LD:  Status of TA 2bd71b48-8000-4834-b93f-1ab567282acd
E/LD:   arch: aarch64
E/LD:  region  0: va 0x40005000 pa 0x0e337000 size 0x002000 flags rw-s (ldelf)
E/LD:  region  1: va 0x40007000 pa 0x0e339000 size 0x009000 flags r-xs (ldelf)
E/LD:  region  2: va 0x40010000 pa 0x0e342000 size 0x001000 flags rw-s (ldelf)
E/LD:  region  3: va 0x40011000 pa 0x0e343000 size 0x004000 flags rw-s (ldelf)
E/LD:  region  4: va 0x40015000 pa 0x0e347000 size 0x001000 flags r--s
E/LD:  region  5: va 0x40016000 pa 0x0e36e000 size 0x001000 flags rw-s (stack)
E/LD:  region  6: va 0x40017000 pa 0x7f8b2c60 size 0x001000 flags rw-- (param)
E/LD:  region  7: va 0x40018000 pa 0x7f8b2a88 size 0x001000 flags rw-- (param)
E/LD:  region  8: va 0x40087000 pa 0x00001000 size 0x01a000 flags r-xs [0]
E/LD:  region  9: va 0x400a1000 pa 0x0001b000 size 0x00c000 flags rw-s [0]
E/LD:   [0] 2bd71b48-8000-4834-b93f-1ab567282acd @ 0x40087000
E/LD:  Call stack:
E/LD:   0x4008f918
E/LD:   0x400876b8
E/LD:   0x4008721c
E/LD:   0x40090894
E/LD:   0x4009066c
E/LD:   0x400876ec
D/TC:? 0 user_ta_enter:201 tee_user_ta_enter: TA panicked with code 0xffff0006
D/TC:? 0 release_ta_ctx:668 Releasing panicked TA ctx
D/TC:? 0 tee_ta_invoke_command:796 Error: ffff3024 of 3
D/TC:? 0 tee_ta_close_session:468 csess 0xe19c860 id 2
D/TC:? 0 tee_ta_close_session:487 Destroy session
D/TC:? 0 destroy_context:326 Destroy TA ctx (0xe19c800)
```

https://optee.readthedocs.io/en/latest/debug/abort_dumps.html

実際に手元のPCで動かしながら聞いて下さい

DEBUG： dumpをからスタックトレースを確認する方法

```
❯ cat dump.txt | ../../optee.4.0.0/optee_os/scripts/symbolize.py -d ./ta/*
E/TC:? 0 TA panicked with code 0xffff0006 (TEE_ERROR_BAD_PARAMETERS)
E/LD:  Status of TA 2bd71b48-8000-4834-b93f-1ab567282acd
E/LD:   arch: aarch64
E/LD:  region  0: va 0x40005000 pa 0x0e337000 size 0x002000 flags rw-s (ldelf)
E/LD:  region  1: va 0x40007000 pa 0x0e339000 size 0x009000 flags r-xs (ldelf)
E/LD:  region  2: va 0x40010000 pa 0x0e342000 size 0x001000 flags rw-s (ldelf)
E/LD:  region  3: va 0x40011000 pa 0x0e343000 size 0x004000 flags rw-s (ldelf)
E/LD:  region  4: va 0x40015000 pa 0x0e347000 size 0x001000 flags r--s
E/LD:  region  5: va 0x40016000 pa 0x0e36e000 size 0x001000 flags rw-s (stack)
E/LD:  region  6: va 0x40017000 pa 0x7f1015f0 size 0x001000 flags rw-- (param)
E/LD:  region  7: va 0x40018000 pa 0x7f101418 size 0x001000 flags rw-- (param)
E/LD:  region  8: va 0x40087000 pa 0x00001000 size 0x01a000 flags r-xs [0] .text .eh_frame .rodata .gnu.hash .ta_head .dynsym .dynstr .hash .rela.dyn
E/LD:  region  9: va 0x400a1000 pa 0x0001b000 size 0x00c000 flags rw-s [0] .dynamic .got .rela.got .data .bss
E/LD:   [0] 2bd71b48-8000-4834-b93f-1ab567282acd @ 0x40087000 (ta/2bd71b48-8000-4834-b93f-1ab567282acd.elf)
E/LD:  Call stack:
E/LD:   0x4008f918 __GP11_TEE_DeriveKey at /mnt/devssd1/miyadev/prog/optee.4.0.0/optee_os/lib/libutee/**********.c:123455
E/LD:   0x400876b8 xxxxxxxxxx ta/**********.c:156
E/LD:   0x4008721c yyyyyyyyyyy at ta/**********.c:82
E/LD:   0x40090894 __ta_invoke_cmd at /mnt/devssd1/miyadev/prog/optee.4.0.0/optee_os/lib/libutee/user_ta_entry_compat.c:95
E/LD:   0x4009066c entry_invoke_command at /mnt/devssd1/miyadev/prog/optee.4.0.0/optee_os/lib/libutee/user_ta_entry.c:382
E/LD:   0x400876ec __ta_entry at /mnt/devssd1/miyadev/prog/optee.4.0.0/optee_os/out/arm/export-ta_arm64/src/user_ta_header.c:51
D/TC:? 0 user_ta_enter:201 tee_user_ta_enter: TA panicked with code 0xffff0006 (TEE_ERROR_BAD_PARAMETERS)
D/TC:? 0 release_ta_ctx:668 Releasing panicked TA ctx
D/TC:? 0 tee_ta_invoke_command:796 Error: ffff3024 of 3
D/TC:? 0 tee_ta_close_session:468 csess 0xe19c860 id 2
D/TC:? 0 tee_ta_close_session:487 Destroy session
D/TC:? 0 destroy_context:326 Destroy TA ctx (0xe19c800)
```

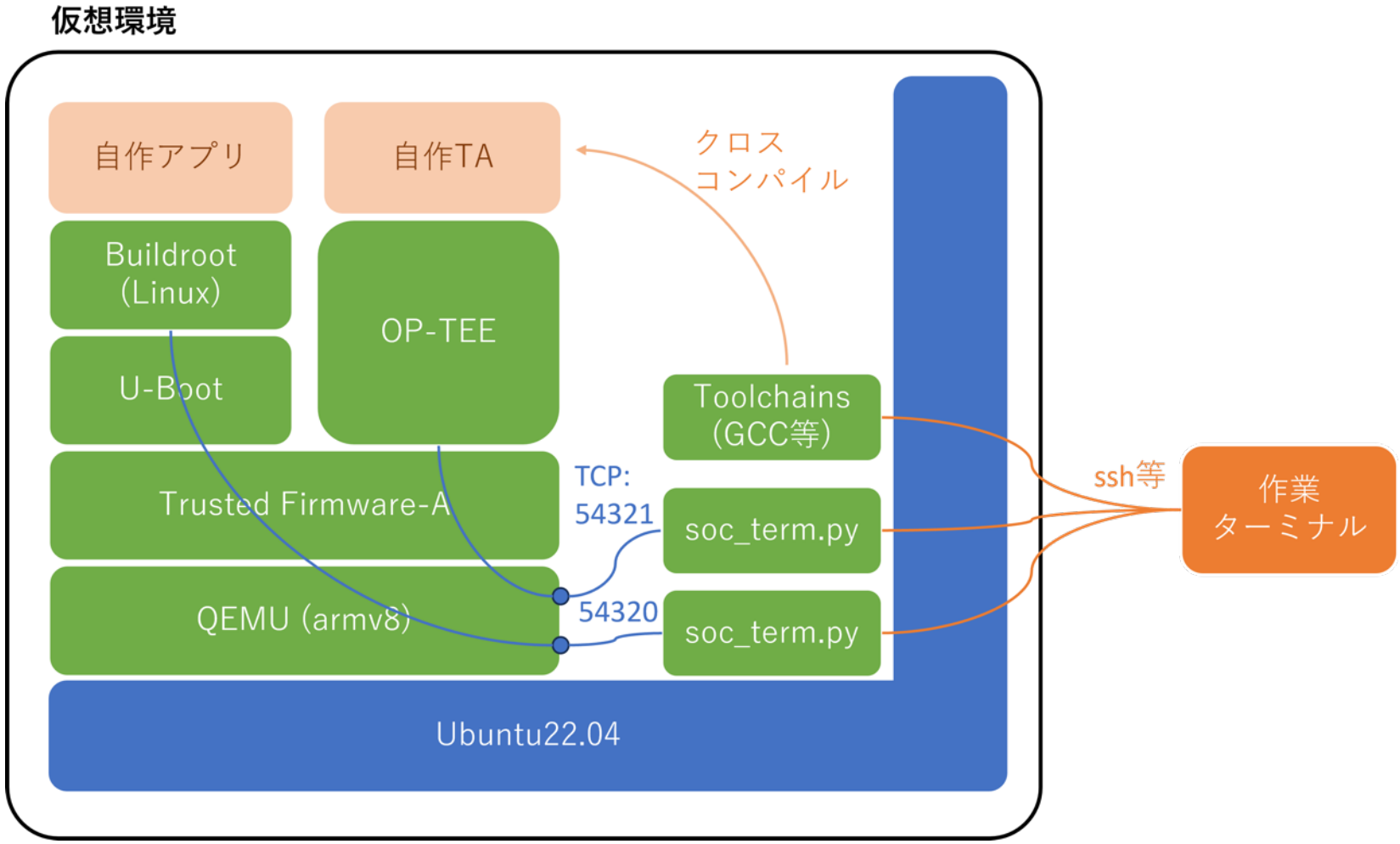
https://optee.readthedocs.io/en/latest/debug/abort_dumps.html

後半まとめ

- **ファイルの内容を確認した**
  - QEMUのホストとなるUbuntu側のファイル（コンパイル環境）
  - QEMU上で動作するLinuxのファイル（実行環境）
  - TEE側のアプリ（Trusted Application）と、それを実行するためのREE側のアプリのソースコード環境
- **実行方法を確認した**
  - GUI
  - CUI
- **プログラムを具体的に追ってみた**
  - TEEとREEが連携するためのフレームワークの確認
  - optee_example/hello_worldのソースコード簡易レビュー
- **デバッグ方法を確認した**
  - GDBによるステップ実行
  - TEE_Panic関数が実行された時にTEE側コンソールに表示されるダンプの分析方法

付録

OP-TEE(QEMUv8版)インストール後のイメージ



最新かつ正確な情報はOP-TEE本家のBuild and RunとDevice specific information(QEMU v8)のページをご覧ください。