Portable TrustZone-Apps mit OP-TEE

Hintergrund, Entwicklung
und Prototyping mit QEMU

Dr.-Ing. Markus Wamser

22. Juli 2020

MIXED MODE

Agenda

01 **OP-TEE und TrustZone**

02 **QEMU und Toolchain**

03 **Trusted Applications I**
Aufbau

04 **Trusted Applications II**
Lebenszyklus

05 **Trusted Applications III**
Hello World im Detail

06 **Trusted Applications IV**
Eine eigene TA

07 **Fazit und Ausblick**

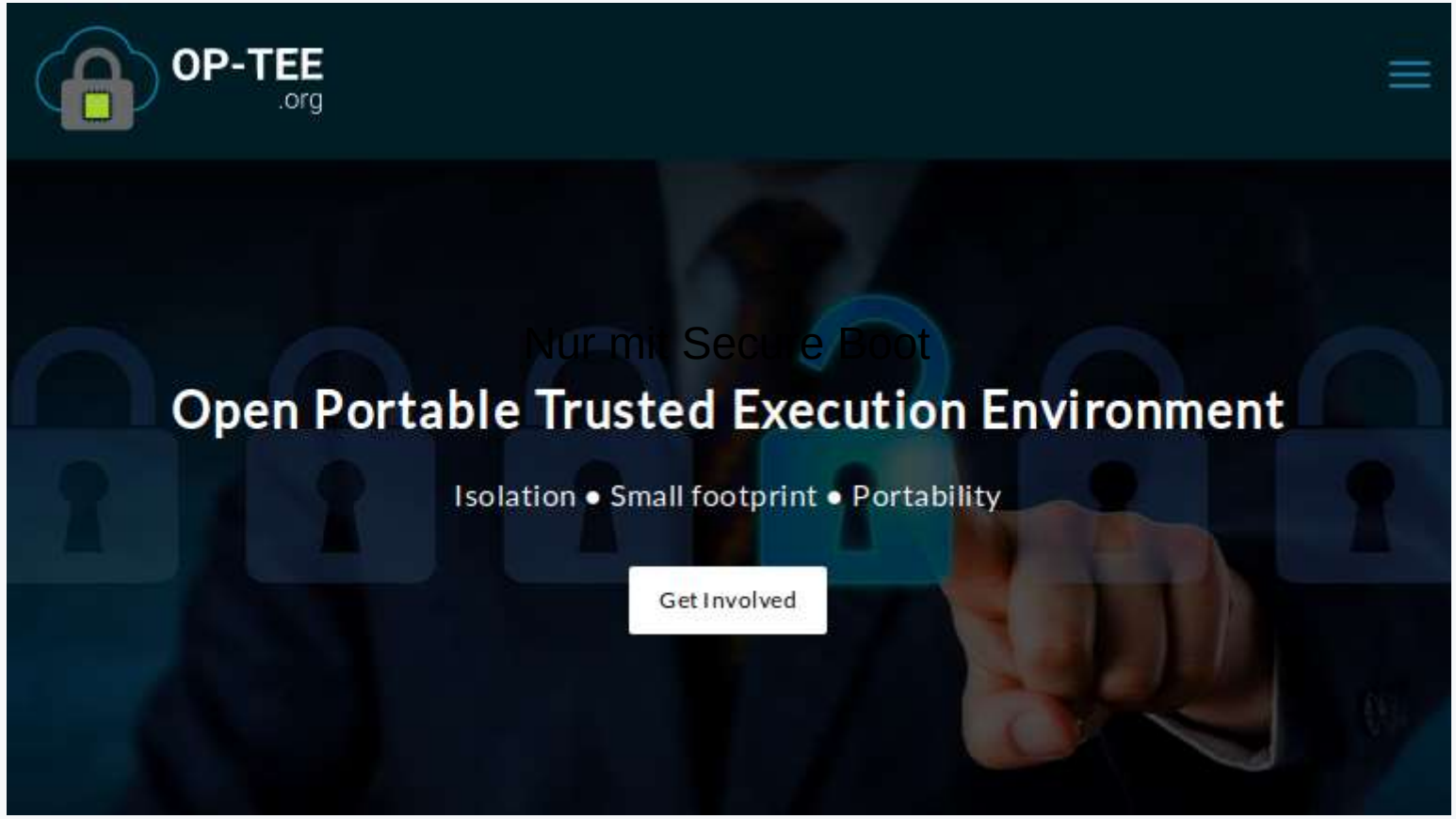
08 **Q & A**

MIXED MODE

OP-TEE

MIXED MODE

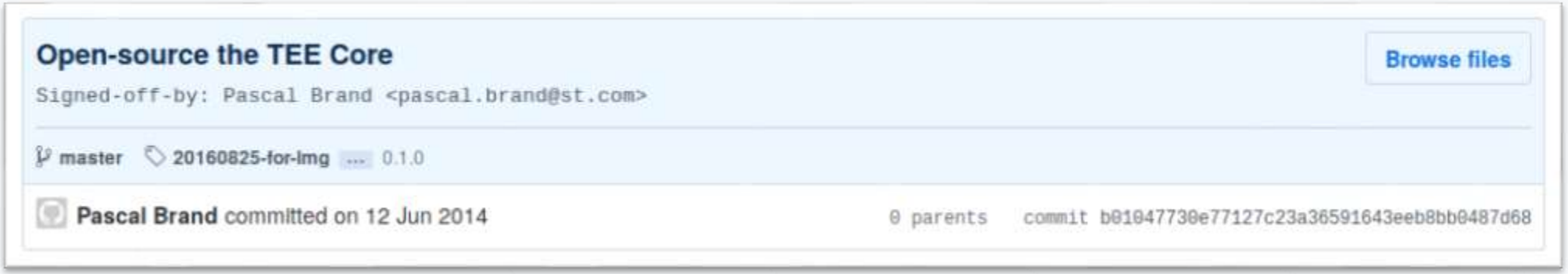
OP-TEE?



Quelle: https://op-tee.org

OP-TEE

- 2004 Support für Arm-Prozessoren mit TrustZone in GCC
- 2009 Standard für TEE-APIs von GlobalPlatform
- 2009 Beginn eines TEE-Frameworks bei ST-Ericcson
- 2013 Zertifiziert durch GlobalPlatform
- 2014 (12. Juni) erster Commit auf GitHub, dann Übergabe an Linaro



- 2017 (2. Juli) TEE Driver im Linux Mainline-Kernel (4.12)

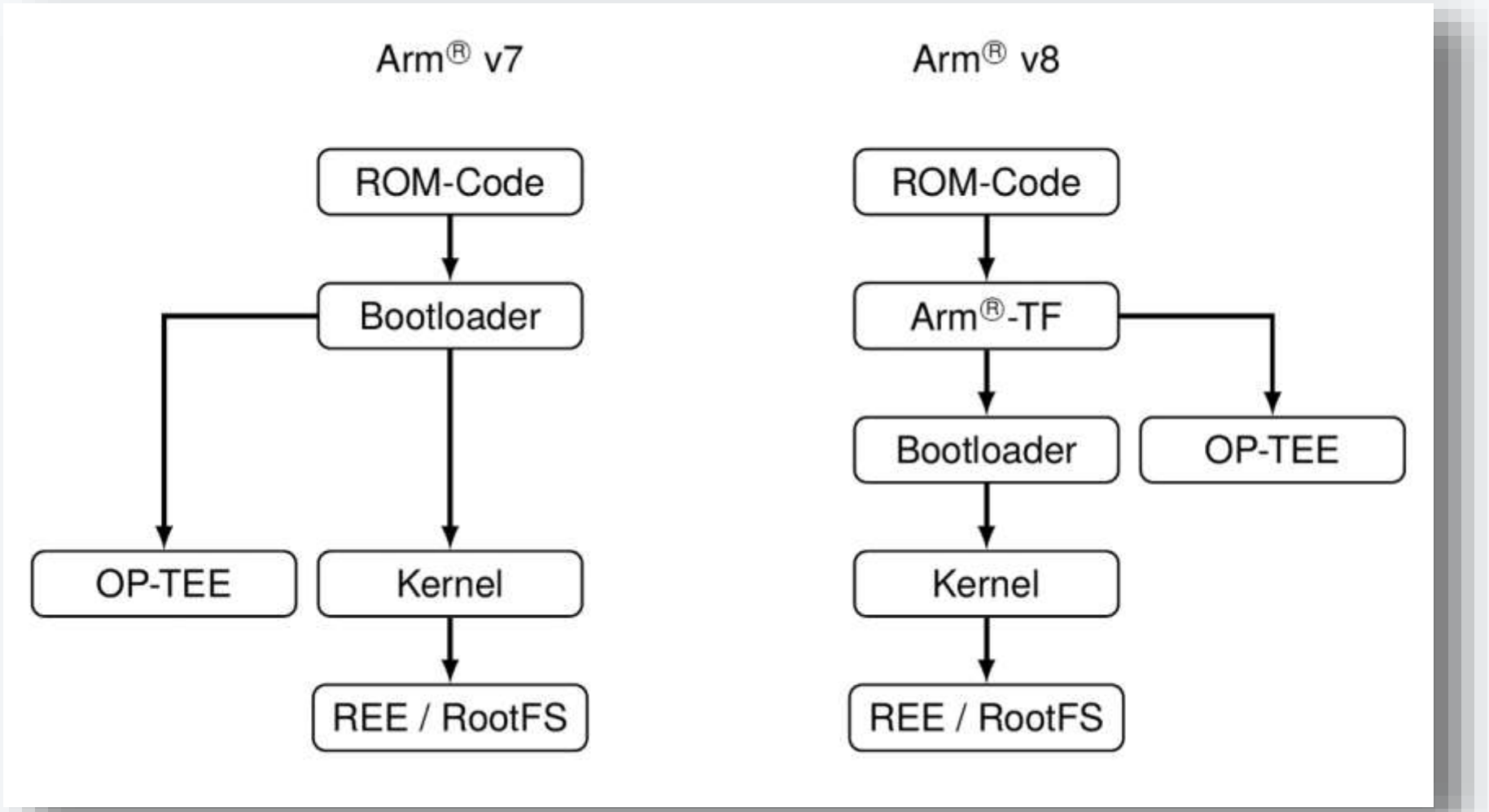
MIXED MODE

OP-TEE

- kleines OS für TrustZone
  - klein: < 500kB
  - aber ohne eigenen Scheduler
  - (noch) keine Threads
  - Hardware-Abstraktion
- bietet komplette GlobalPlatform TEE-API + Extensions
  - Anwendungen können auch gegen andere TEE kompiliert werden
  - Kryptobeschleuniger (z.B CAAM) für Anwendung transparent eingebunden
- führt nur verifizierten Code aus
  - Vertrauenskette vom Power-On bis zur Trusted App

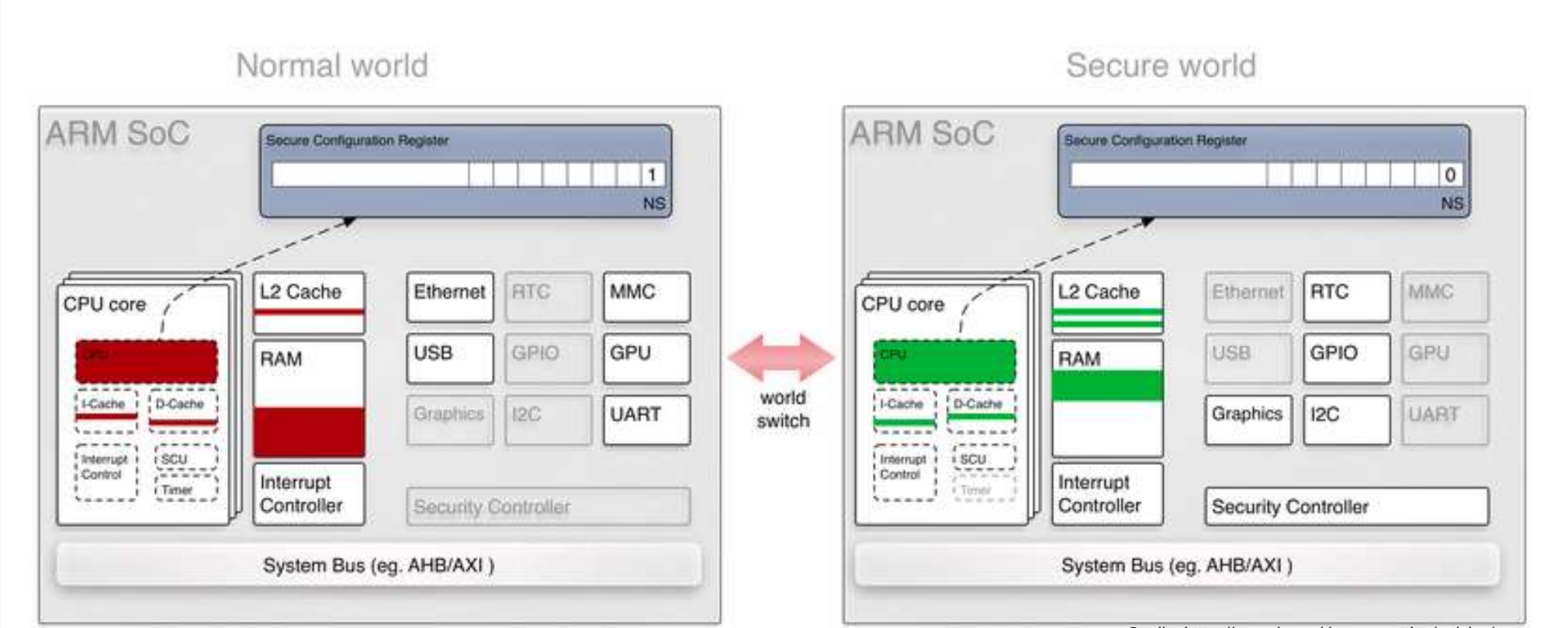
OP-TEE

Secure Boot



TrustZone

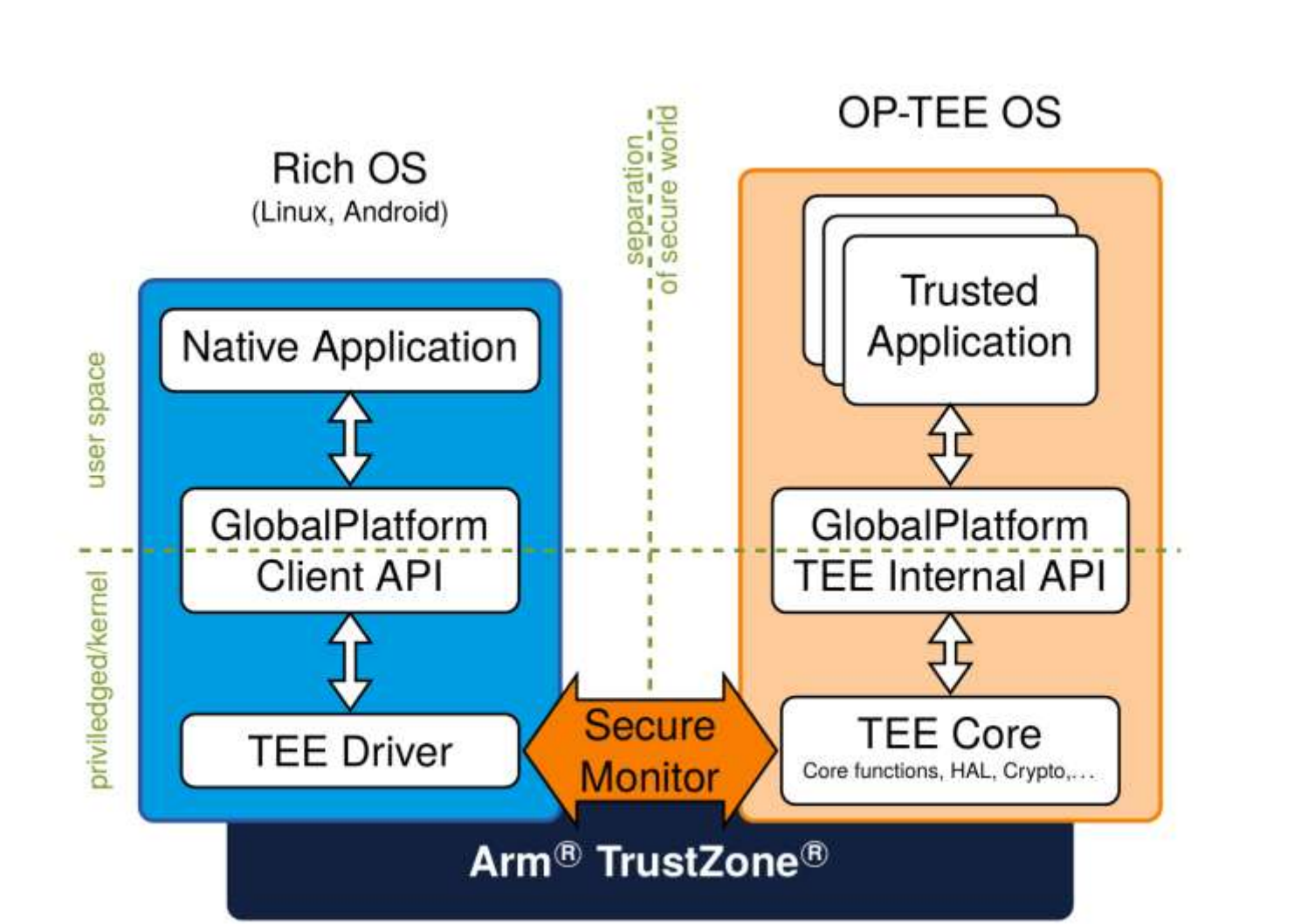
Architektur



Quelle: https://genode.org/documentation/articles/trustzone

OP-TEE

Architektur



nach: https://www.linaro.org/blog/op-tee-open-source-security-mass-market/

TrustZone & QEMU

Spielwiese

- QEMU unterstützt seit 2014 Emulation der TrustZone

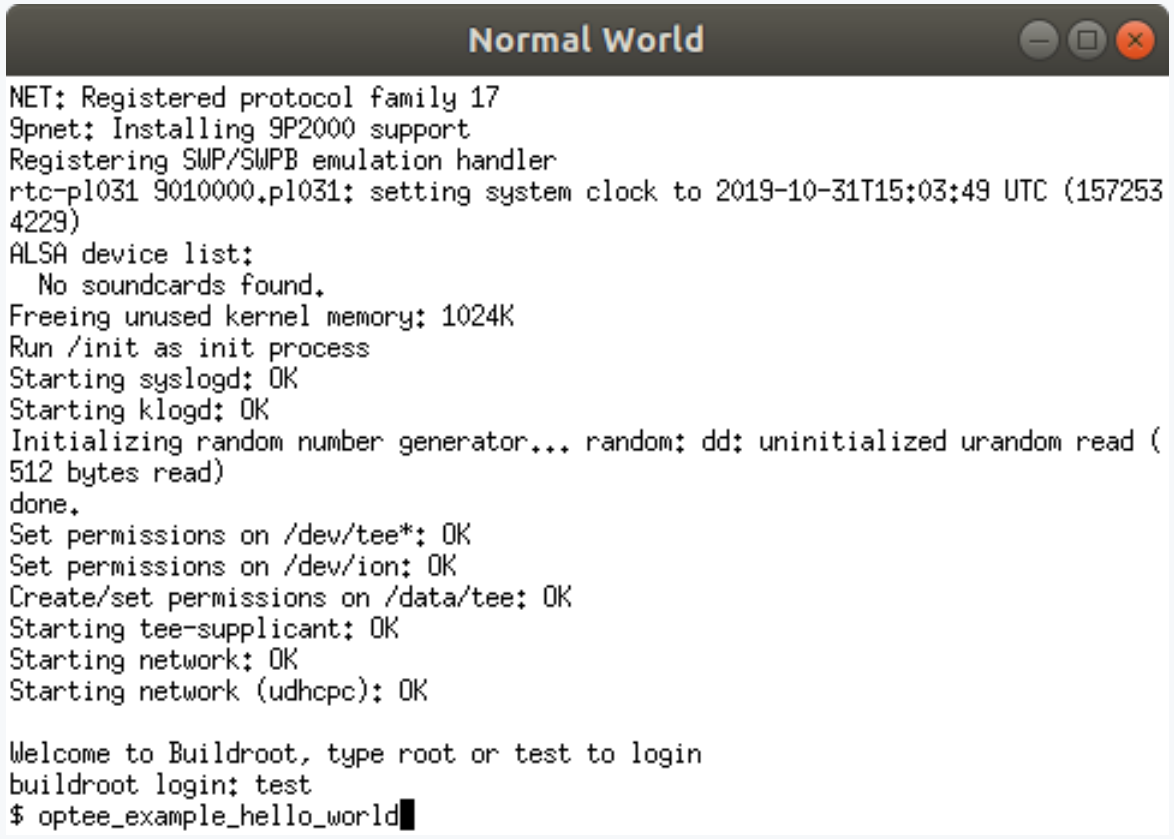
- Manifest für *repo* mit allen Quellen (QEMU, OP-TEE, U-Boot, Buildroot)
  - https://github.com/OP-TEE/manifest/blob/master/qemu_v8.xml
  - Manifeste für andere Plattformen

```
arm-trusted-firmware      optee_client      # Client API
buildroot                 optee_examples    # Beispiele
qemu                      optee_benchmark   # Benchmarks
linux                     optee_os          # OS & Core API
toolchains                optee_test        # xtest / Regressionstests
u-boot
                          build             # Arbeitsverzeichnis
```

Start

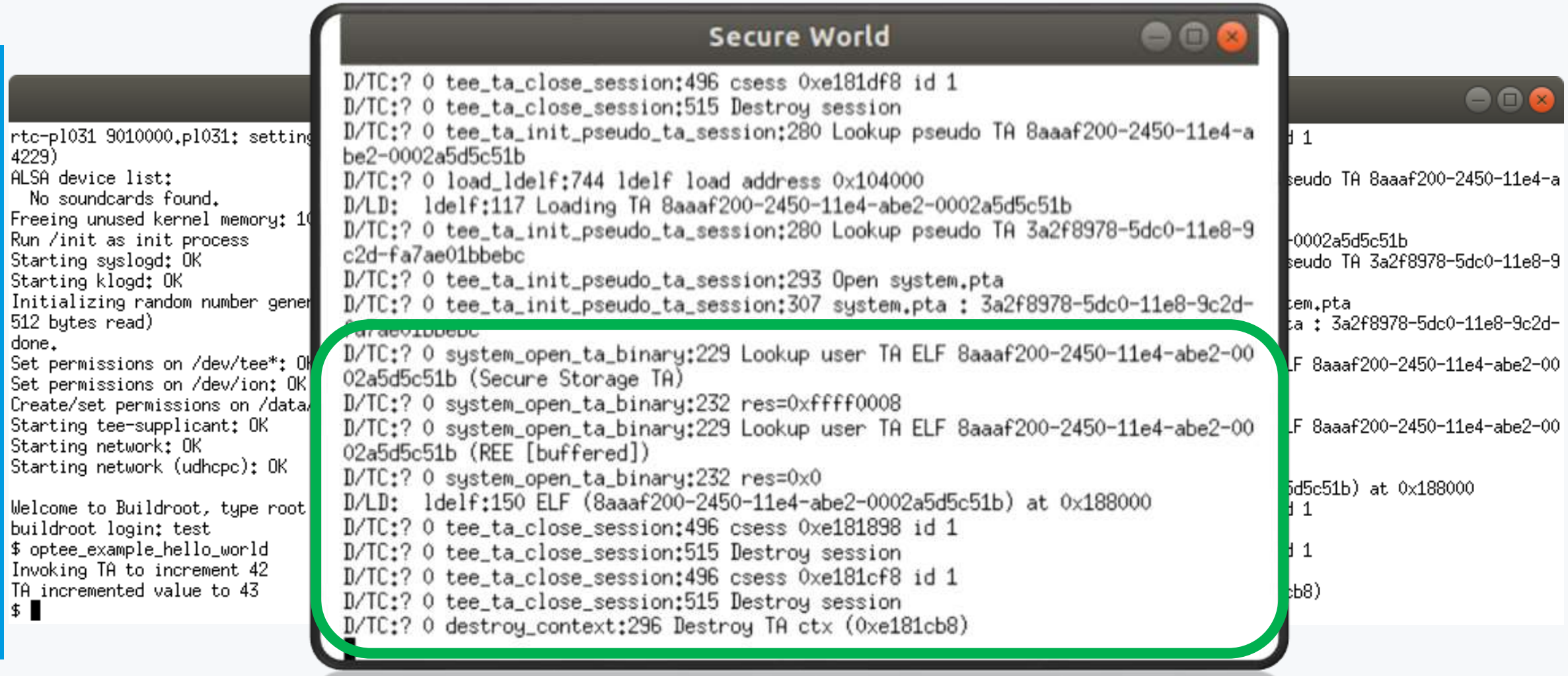
MIXED MODE

Hello World



MIXED MODE

Hello World


rtc-pl031 9010000.pl031: setting
4229)
ALSA device list:
No soundcards found.
Freeing unused kernel memory: 1
Run /init as init process
Starting syslogd: OK
Starting klogd: OK
Initializing random number gener
512 bytes read)
done.
Set permissions on /dev/tee*: O
Set permissions on /dev/ion: OK
Create/set permissions on /data
Starting tee-supplicant: OK
Starting network: OK
Starting network (udhcpc): OK
Welcome to Buildroot, type root
buildroot login: test
$ optee_example_hello_world
Invoking TA to increment 42
TA incremented value to 43
$
Secure World
D/TC:? 0 tee_ta_close_session:496 csess 0xe181df8 id 1
D/TC:? 0 tee_ta_close_session:515 Destroy session
D/TC:? 0 tee_ta_init_pseudo_ta_session:280 Lookup pseudo TA 8aaaf200-2450-11e4-a
be2-0002a5d5c51b
D/TC:? 0 load_ldelf:744 ldelf load address 0x104000
D/LD: ldelf:117 Loading TA 8aaaf200-2450-11e4-abe2-0002a5d5c51b
D/TC:? 0 tee_ta_init_pseudo_ta_session:280 Lookup pseudo TA 3a2f8978-5dc0-11e8-9
c2d-fa7ae01bbebc
D/TC:? 0 tee_ta_init_pseudo_ta_session:293 Open system.pta
D/TC:? 0 tee_ta_init_pseudo_ta_session:307 system.pta : 3a2f8978-5dc0-11e8-9c2d-
D/TC:? 0 system_open_ta_binary:229 Lookup user TA ELF 8aaaf200-2450-11e4-abe2-00
02a5d5c51b (Secure Storage TA)
D/TC:? 0 system_open_ta_binary:232 res=0xffff0008
D/TC:? 0 system_open_ta_binary:229 Lookup user TA ELF 8aaaf200-2450-11e4-abe2-00
02a5d5c51b (REE [buffered])
D/TC:? 0 system_open_ta_binary:232 res=0x0
D/LD: ldelf:150 ELF (8aaaf200-2450-11e4-abe2-0002a5d5c51b) at 0x188000
D/TC:? 0 tee_ta_close_session:496 csess 0xe181898 id 1
D/TC:? 0 tee_ta_close_session:515 Destroy session
D/TC:? 0 tee_ta_close_session:496 csess 0xe181cf8 id 1
D/TC:? 0 tee_ta_close_session:515 Destroy session
D/TC:? 0 destroy_context:296 Destroy TA ctx (0xe181cb8)

Hello World

Typische Codestruktur

```
optee_examples> tree hello_world/
hello_world/
├── Android.mk
├── CMakeLists.txt
├── host                                # REE-Applikation
│   ├── main.c
│   └── Makefile
├── Makefile
└── ta                                  # TEE-Applikation
    ├── Android.mk
    ├── hello_world_ta.c
    ├── include
    │   └── hello_world_ta.h            # User-defined API REE <-> TEE
    ├── Makefile
    ├── sub.mk
    └── user_ta_header_defines.h        # TEE-Konfiguration (Stack, Heap, Meta)
```

TA Lifecycle

(k)ein Drama in 5 Akten

- Feste API der TA

```
TA_CreateEntryPoint()                # Instanziierung
                                     |
                                     V
TA_OpenSessionEntryPoint()           # Sitzungsaufbau REE <-> TEE
                                  ___|
                                 /   V
TA_InvokeCommandEntryPoint()     |   # RPCs mit TA-spezifischer API
                                 \___|
                                     V
TA_CloseSessionEntryPoint()          # Sitzungsabbau
                                     |
                                     V
TA_DestroyEntryPoint()               # Clean Up
```

Inside Hello World

Ein Spaziergang – REE main.c

```
#include <tee_client_api.h>                          // OP-TEE TEE client API
#include <hello_world_ta.h>                          // UUID + API der TA

int main(void)
{
        TEEC_Result res; TEEC_Context ctx;
        TEEC_Session sess; TEEC_Operation op;        // … weitere Variablen

        res = TEEC_InitializeContext(NULL, &ctx);    // Session Context
        res = TEEC_OpenSession(&ctx, &sess, &uuid,
                               TEEC_LOGIN_PUBLIC, NULL, NULL, &err_origin);

        memset(&op, 0, sizeof(op));                  // RPC vorbereiten
        op.paramTypes = TEEC_PARAM_TYPES(TEEC_VALUE_INOUT, TEEC_NONE,
                                         TEEC_NONE, TEEC_NONE);
        op.params[0].value.a = 42;
```

Inside Hello World

Ein Spaziergang – REE main.c

```
        // eigentlicher RPC
        res = TEEC_InvokeCommand(&sess, TA_HELLO_WORLD_CMD_INC_VALUE, &op,
                                 &err_origin);

        TEEC_CloseSession(&sess);            // Session abbauen

        TEEC_FinalizeContext(&ctx);          // Context aufräumen

        return 0;
}
```

Inside Hello World

Ein Spaziergang – Bindeglied hello_world_ta.h

```
#ifndef TA_HELLO_WORLD_H
#define TA_HELLO_WORLD_H

#define TA_HELLO_WORLD_UUID \
        { 0x8aaaf200, 0x2450, 0x11e4, \
                { 0xab, 0xe2, 0x00, 0x02, 0xa5, 0xd5, 0xc5, 0x1b} }

/* The function IDs implemented in this TA */
#define TA_HELLO_WORLD_CMD_INC_VALUE        0
#define TA_HELLO_WORLD_CMD_DEC_VALUE        1

#endif /*TA_HELLO_WORLD_H*/
```

Inside Hello World

Ein Spaziergang – TEE hello_world_ta.c

```
#include <tee_internal_api.h>
#include <tee_internal_api_extensions.h>

#include <hello_world_ta.h>

TEE_Result TA_CreateEntryPoint(void)        // neue Instanz
{
        return TEE_SUCCESS;
}

void TA_DestroyEntryPoint(void)             // Teardown
{
        // NOP
}
```

Inside Hello World

Ein Spaziergang – TEE hello_world_ta.c

```
TEE_Result TA_OpenSessionEntryPoint(uint32_t param_types,
                TEE_Param __maybe_unused params[4],
                void __maybe_unused **sess_ctx)
{
        uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE);
        if (param_types != exp_param_types)
                return TEE_ERROR_BAD_PARAMETERS;

        /* If return value != TEE_SUCCESS the session will not be created. */
        return TEE_SUCCESS;
}

void TA_CloseSessionEntryPoint(void __maybe_unused *sess_ctx) {}  // NOP
```

Inside Hello World

Ein Spaziergang – TEE hello_world_ta.c

```
static TEE_Result inc_value(uint32_t param_types,
        TEE_Param params[4])
{
        uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INOUT,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE);
        if (param_types != exp_param_types)
                return TEE_ERROR_BAD_PARAMETERS;
        params[0].value.a++;
        return TEE_SUCCESS;
}


static TEE_Result dec_value(…)      // entsprechend mit params[0].value.a--;
```

Inside Hello World

Ein Spaziergang – TEE hello_world_ta.c

```
TEE_Result TA_InvokeCommandEntryPoint(void __maybe_unused *sess_ctx,
                      uint32_t cmd_id,
                      uint32_t param_types, TEE_Param params[4])
{

        switch (cmd_id) {                                   // RPC dispatch
        case TA_HELLO_WORLD_CMD_INC_VALUE:
                return inc_value(param_types, params);
        case TA_HELLO_WORLD_CMD_DEC_VALUE:
                return dec_value(param_types, params);
        default:
                return TEE_ERROR_BAD_PARAMETERS;
        }
}
```

Hello SAEC-Days

A new app is born

- Vorgehen
  - Kopie des hello_world-Beispiels anlegen & Strings/Dateinamen anpassen
  - Neue UUID erzeugen
  - (manueller) Regressionstest
  - neue Funktionalität
- Konzept der Beispiel-App
  - Basierend auf hello_world-Beispiel
  - One-Time PIN (wie für 2FA)
  - einfacher (und unsicherer) als HOTP-Beispiel ☺

Hello SAEC-Days

A new app is born

- Schritt-für-Schritt-Erklärung in Mini-Workshop (auf Anfrage)
- Jetzt nur die Highlights!

MIXED MODE

Hello SAEC-Days

TEE hello_xxx_ta.c (Funktionalität - Helper)

```
static TEE_Result hashpin(const uint8_t pin[6], const uint8_t salt[32],
                          uint8_t hash[32])
{
        TEE_OperationHandle op_handle = TEE_HANDLE_NULL;
        TEE_Result res = TEE_SUCCESS;

        res = TEE_AllocateOperation(&op_handle, TEE_ALG_SHA256,
                                    TEE_MODE_DIGEST, 0);

        size_t actualhashLen = 32;
        TEE_DigestUpdate(op_handle, pin, 6);
        res = TEE_DigestDoFinal(op_handle, salt, 32, hash, &actualhashLen);

        if (op_handle != TEE_HANDLE_NULL) TEE_FreeOperation(op_handle);
        return res;
}
```

Hello SAEC-Days

TEE hello_xxx_ta.c (Funktionalität – Session I)

```
struct pinsession {
    uint8_t hashedPIN[32]; /**< \brief hashed value of secret PIN */
    uint8_t sessionSalt[32]; /**< \brief salt used for hashing */
};

TEE_Result TA_OpenSessionEntryPoint(uint32_t param_types,
                TEE_Param __maybe_unused params[4],
                void __maybe_unused **sess_ctx)
{
        uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE);

    struct pinsession *state; /**< \brief (secret) session state */
```

MIXED MODE

Hello SAEC-Days

TEE hello_xxx_ta.c (Funktionalität – Session III)

```
void TA_CloseSessionEntryPoint(void __maybe_unused *sess_ctx)
{
        struct pinsession *state = sess_ctx;
        TEE_Free(state);
}
```

Ausblick

MIXED MODE

Andere TEEs (Auswahl)

(teilweise) kompatibel zu OP-TEE

- Kommerzielle Implementierungen
  - Qualcomm QSEE
  - Huawei Hisilicon TEE
- Akademische Implementierungen
  - Contego-TEE
  - MicroTEE
  - ANDIX OS
  - OpenTEE

Und weiter?

- RTFM: https://optee.readthedocs.io/en/latest/
- Weitere Beispiele in optee_examples (Auswahl)
  - RSA
  - AES
  - HOTP
  - Secure Storage
- Real-World Usecase: IP-Schutz/Prozesssteuerung
  - Hybride Verschlüsselung (RSA+AES-GCM)
  - Private-Key über Fused Key geschützt, nur in TA sichtbar
  - Mocking für Fused Key ermöglichte (fast) komplette Entwicklung ohne Ziel-HW

Fazit

- Perfekt sicher?
  - Nein: TAs sind nur Software -> Schwachstellen (siehe: Qualcomm Key leaks)
    mehr Beispiele: https://github.com/enovella/TEE-reversing
  - aber gutes Preis/Leistung-Verhältnis
- schnelle Entwicklung dank standardisierter API
  - kurze Develop-Test Zyklen dank Emulation
  - früherer Entwicklungsstart dank Emulation, parallel zur HW-Entwicklung
- portabel dank standardisierter API
  - „Develop once, run everywhere“

Fragen?

Antworten.

Dr.-Ing.
Markus Wamser
Expert Embedded Security

Markus.Wamser@mixed-mode.de
Tel.: 089/8 98 68-200
www.mixed-mode.de