

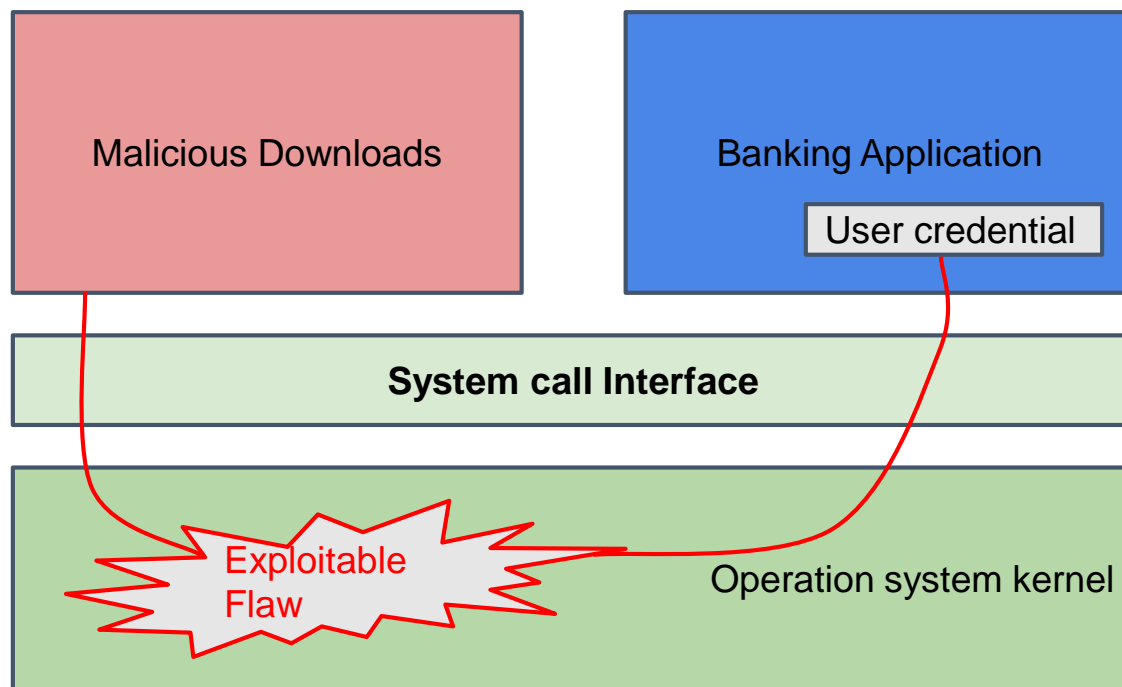
Trusted Execution Environment 2

Jinsoo Jang

ARM TrustZone

Need for Security

- Valuable data, such as banking credentials
- Downloading arbitrary third-party apps.



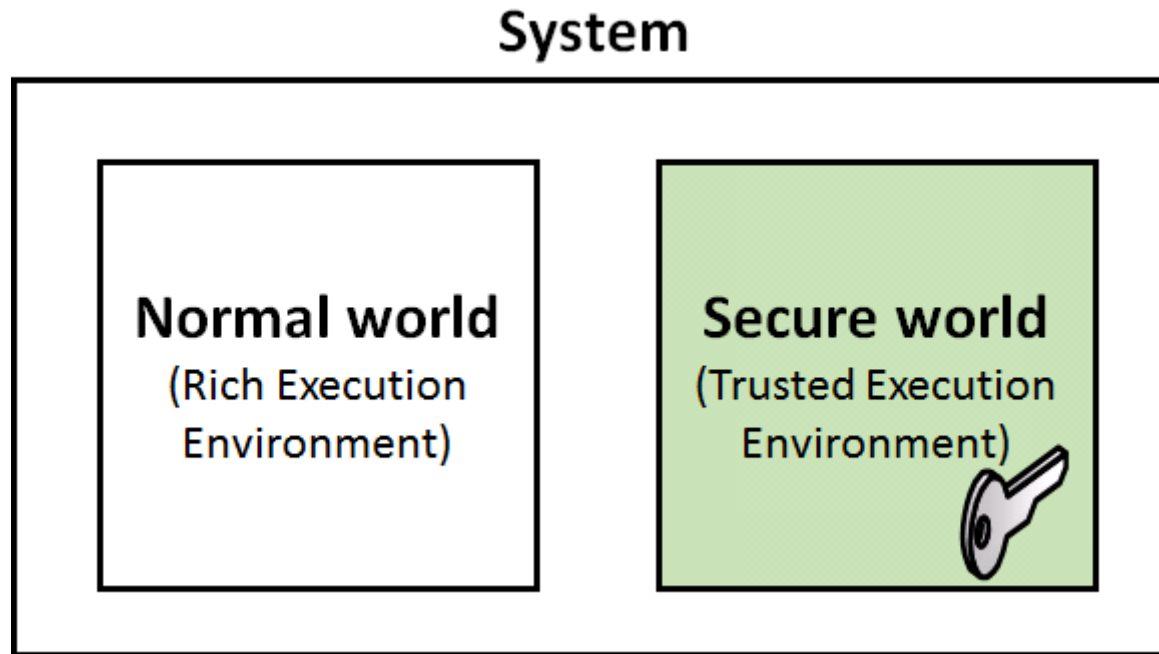
How are Devices Attacked?

- Hack attack
 - Software level attack
 - Malwares
- Shack attack
 - Low-budget hardware attack
 - e.g. Using a JTAG debugger
- Lab attack
 - Laboratory equipments
 - Electron microscopes
 - Transistor-level reversing
 - Power usage analysis



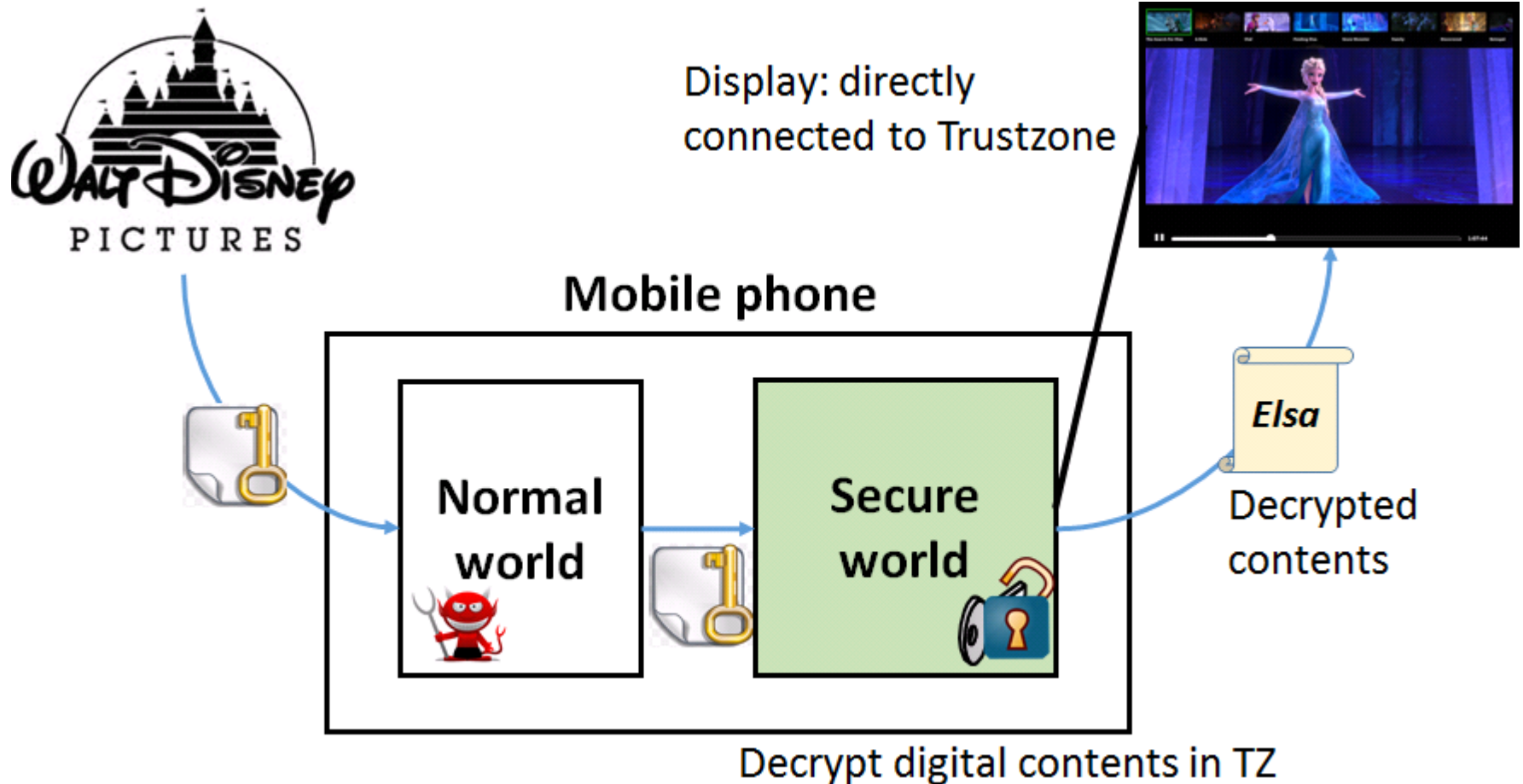
ARM TrustZone (Overview)

- Extension to the processors
- Hardware security
- Countermeasure for may of hack & shack attacks

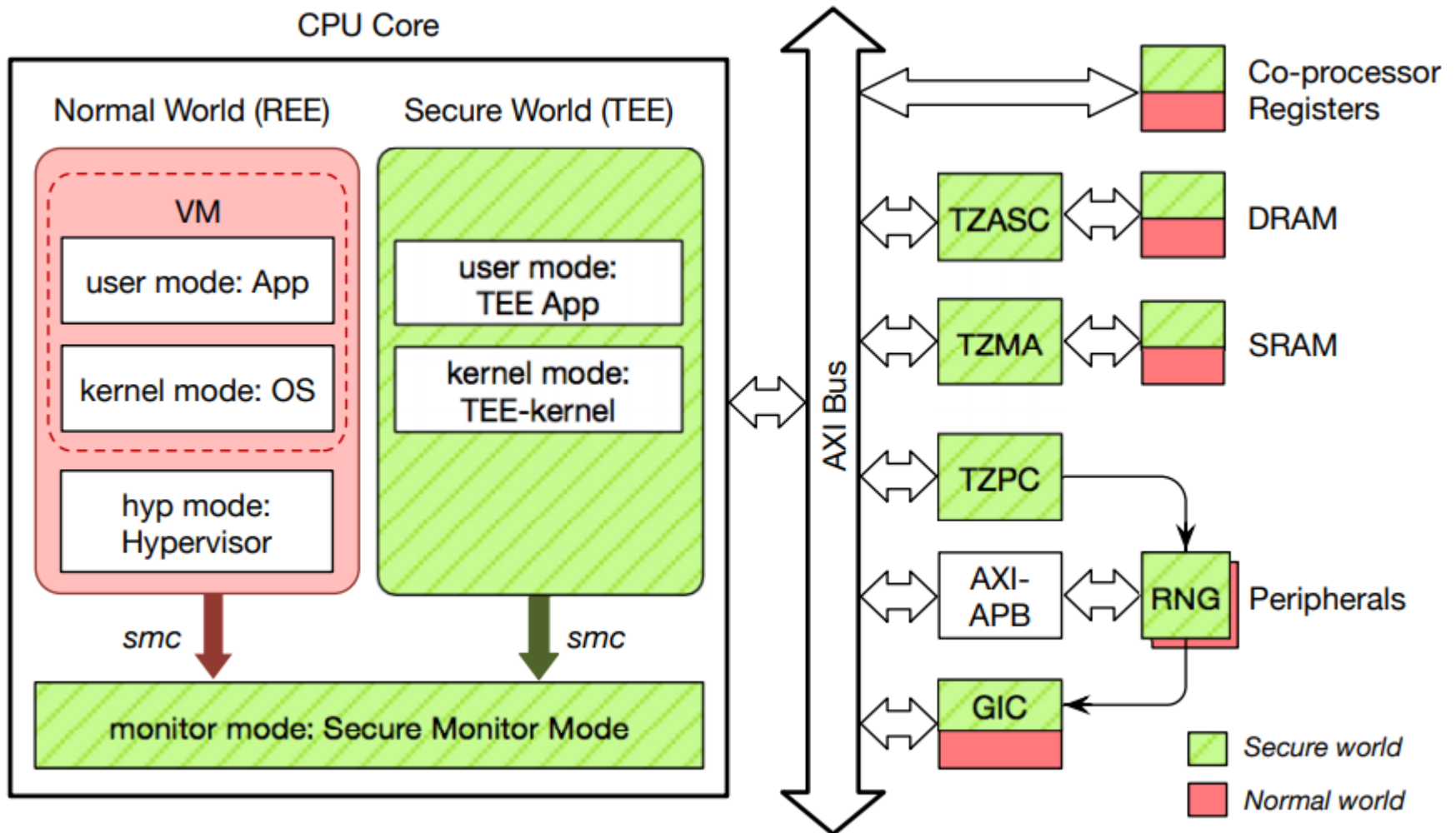


Example Use Case

- DRM (Digital Right Management) in TZ



TrustZone Hardware Architecture

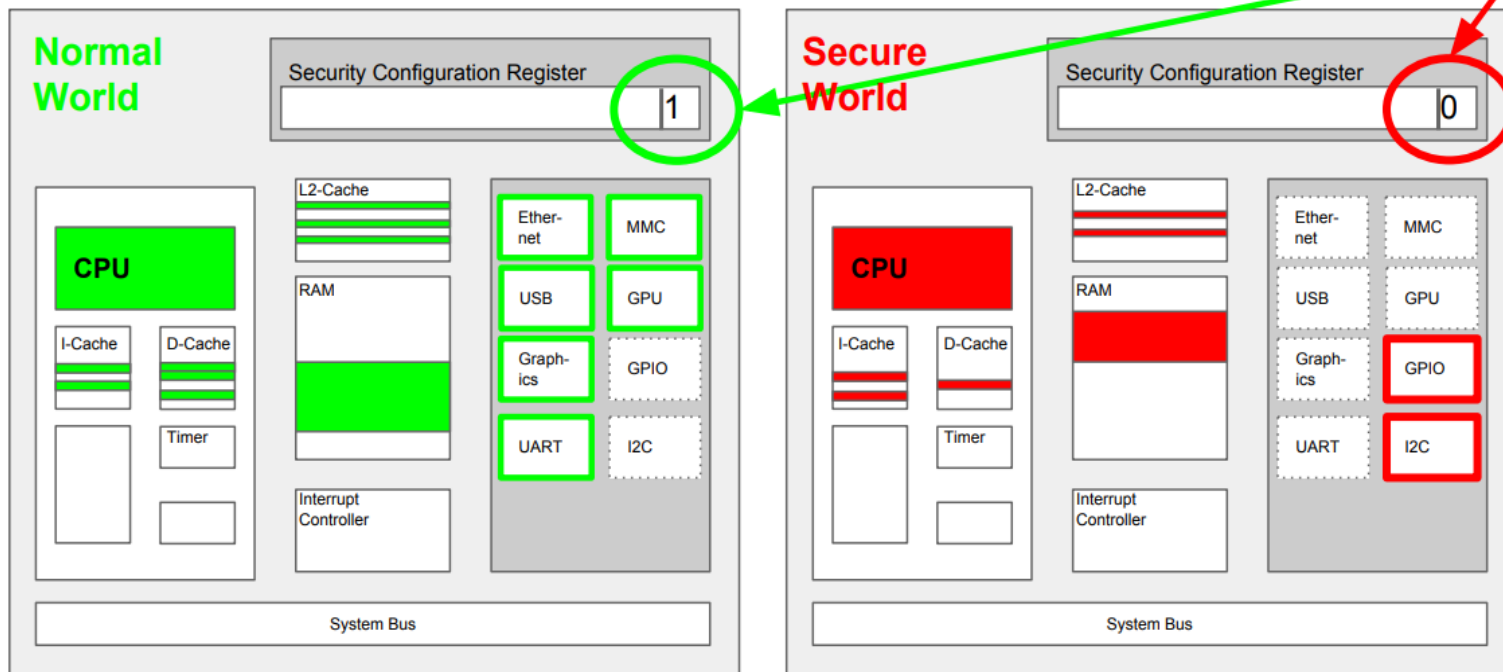


Partions SoC into Normal and Secure

ARM TrustZone®

Partitions SoC into **Normal World** and **Secure World**

Non-Secure (NS) bit (bit 0) in the Secure Configuration Register in CP15



TrustZone to TEE in 3 Steps

1. Define secure hardware architecture
 - Two separate domains: normal and secure
 - Extends across system
 - processor, display, keypad, memory, clock, radios
2. Implement in silicon system on chip (SoC)
 - Enforcing secure/normal separation physically
3. Combine SoC with Trusted SW
 - Separate but connected to main OS

Result:

A **T**rusted **E**xecution **E**nvironment (TEE)

1. Ready to develop and run trusted services



Development env. & SierraTEE Example

Open source TZ software

- Open Virtualization
 - Provides TrustZone OS and toolchains
 - Download:
 - <http://www.openvirtualization.org/download.html>

SierraTEE and SierraVisor Source Code	
License:	GPLv2
Requirements:	Linux 2.6 (or above) development platform, JDK 6.0
Open Virtualization SDK:	SDK_Sep25_2014_TEE.tar.gz
Toolchain:	toolchain_sep_25_2014.tar.bz2 (99.9 MB)
Date of SDK Release:	September 25, 2014
64-bit Rootfs	rootfs.bz2
Documentation:	Open Virtualization Developer's Guide, Build Instructions, and Porting Guide

SierraVisor and SierraTEE Binary for Xilinx(R) Zynq-7000 AP SOC	
License:	Free
ZC702 Binary:	xilinx_zc702_Jan02_2013.tar.bz2 (35.4 MB)
Date of Binary Release:	January 2, 2013
Documentation:	Xilinx Boot Guide

ARM Development Tools

- Fast Models
 - ARM instruction set simulator
 - Provides a debugging environment
 - Download: www.arm.com

The screenshot displays the ARM Fast Models development environment, which includes the System Canvas, Model Debugger, and FVP terminal.

System Canvas (FVP_Ve_Cortex-A15x1.sgproj) - FVP_Ve_Cortex-A15x1.lisa

The System Canvas shows a block diagram of the FVP_Ve_Cortex-A15x1.lisa system. The components list includes:

Component Name	Version	Type	File
AMBAPV2PVBus	8.3.64	Bus	/home/jisjang/...
AMBAPVSignal2SGSignal	8.3.64	Other	/home/jisjang/...
AMBAPVSignal2SGStateSignal	8.3.64	Other	/home/jisjang/...
AMBAPVValue2SGValue64	8.3.64	Other	/home/jisjang/...
AMBAPVValue2SGValueState64	8.3.64	Other	/home/jisjang/...
ARM926CT	8.3.64	Core	/home/jisjang/...
ARM968CT	8.3.64	Core	/home/jisjang/...
ARM1136CT	8.3.64	Core	/home/jisjang/...
ARM1176CT	8.3.64	Core	/home/jisjang/...
ARMAEM6AMPCT	8.3.64	Core	/home/jisjang/...
ARMAEM6AMPx1CT	8.3.64	Core	/home/jisjang/...
ARMAEM6AMPx4CT	8.3.64	Core	/home/jisjang/...

The Model Debugger (cluster.cpu0) shows the source code for dispatcher_task.c, which includes a while loop and a conditional statement. The assembly view shows the corresponding instructions, including MOV, LDR, STR, and ADD. The Register view shows the current state of the registers, including R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, and the MODE register.

FVP terminal_0

The FVP terminal shows the output of the system initialization, including the creation of the GlobalIOL for the user and the dispatch task id 0x11.

Local Variable/Parameter

Local Variable/Parameter	Type	Value
cur_task_id	int	0x00000111 (273)
ret_val	int	0x00000000 (0)
svc_id	int	0x00000004 (4)
task_id	int	0x00000112 (274)
cmd_id	int	0x00000011 (17)
cmd_type	int	0x00000001 (1)
amount_fault...	int	0xC73AD1D0 (-946155056)
amount_fault...	int	0x00000000 (0)
amount_desc...	int	0x00000000 (0)

Source breakpoint is hit

Warning (2001) Source file 'sw_semaphore_asm.s' not found. Do you want to search for Source breakpoint is hit - home/jisjang/Downloads/trustzone/sdks/trustzone/t...

Sample TrustZone Application

- Running a sample crypto program
 - **Encrypts & decrypts a input Sting by using TrustZone**

```
ed
egrep
false
fgrep
grep
# ./otzapp.elf
Input plain text: HELLO NETSEC2015
Input plain text in Hex:0x48 0x45 0x4c 0x4c 0x4f 0x20 0x4e 0x45 0x54 0x53 0x45 0
x43 0x32 0x30 0x31 0x35
-----Now performing AES-128-CBC encryption-----
device closed
AES-128-CBC encryption test passed
Encrypted text in Hex: 0x3b 0x9b 0xc6 0x9a 0xd5 0x46 0xe8 0xb5 0x5e 0xdb 0x9d 0x
9b 0x93 0x70 0x1e 0x54
-----Now performing AES-128-CBC decryption-----
device closed
AES-128-CBC decryption test passed
Decrypted text:HELLO NETSEC2015
Decrypted text in Hex: 0x48 0x45 0x4c 0x4c 0x4f 0x20 0x4e 0x45 0x54 0x53 0x45 0x
43 0x32 0x30 0x31 0x35
#
```

Input string:

"HELLO NETSEC2015"

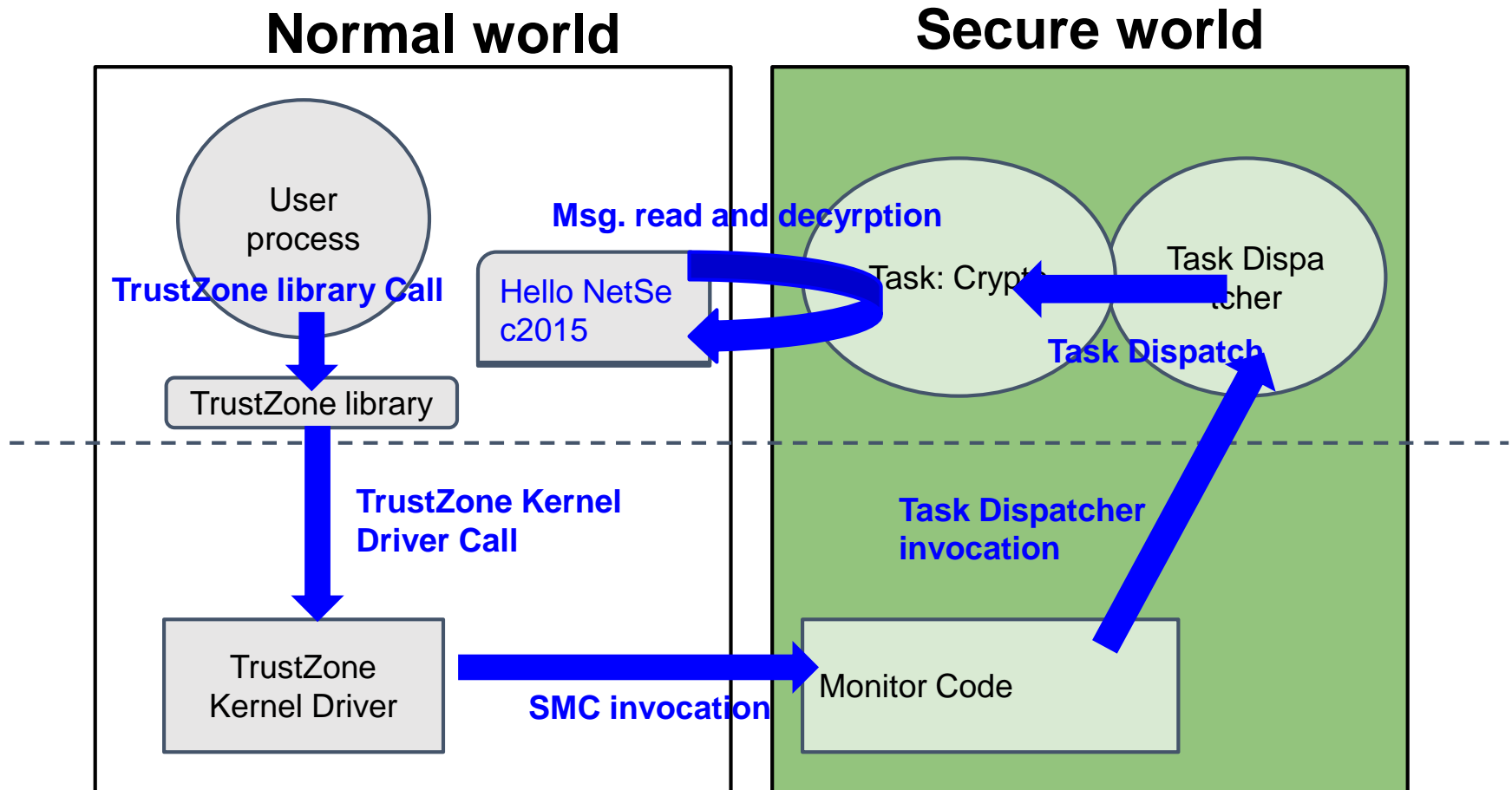
Encrypted string in hex:

0x3b0x9b0xc6.....

Restoring the string by decryptio
n

How It Works? (Overview)

- Data decryption example



How It Works? (Source Code)

- Client application (User mode in Normal world)
 - requests crypto service to TrustZone

Open TrustZone kernel driver

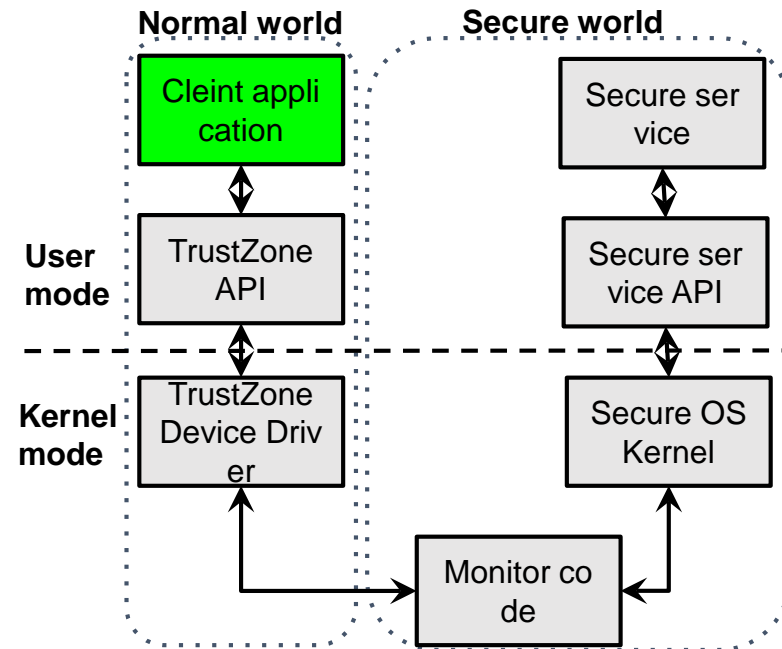
```
perform_crypto(unsigned char *input_buf,int input_buf_len,
               unsigned char *init_vector,int init_vector_len,
               unsigned char *key_buf,int key_len,int cmd_type,
               unsigned char *output_buf, int *output_buf_len,
               unsigned char cipher_action)
```

```
crypto_data_t crypt_data;
uint8_t *out_data,in_data[1];
otz_device_t device_otz;
otz_session_t session_otz;
otz_operation_t operation_otz;
otz_return_t ret=0, service_ret;
```

```
device_otz.ui_state = OTZ_STATE_UNDEFINED;
ret = otz_device_open("/dev/otz_client", (void*)0_RDWR, &device_otz);
if (ret != OTZ_SUCCESS){
    perror("device open failed\n");
    return 0;
}
```

Invoke TrustZone API

```
on_otz.ui_state = OTZ_STATE_UNDEFINED;
operation_otz.ui_state = OTZ_STATE_UNDEFINED;
ret = otz_operation_prepare_open(&device_otz,OTZ_SVC_CRYPT, NULL, NULL,
                                &session_otz, &operation_otz);
if (ret != OTZ_SUCCESS) {
    goto end_func;
}
/* Call otz_operation_perform to open session */
ret = otz_operation_perform(&operation_otz, &service_ret);
if (ret != OTZ_SUCCESS){
    if (ret == OTZ_ERROR_SERVICE)
        printf("%s \n",otz_strerror(ret));
    else
        perror("session open failed\n");
    session_otz.ui_state = OTZ_STATE_UNDEFINED;
}
```



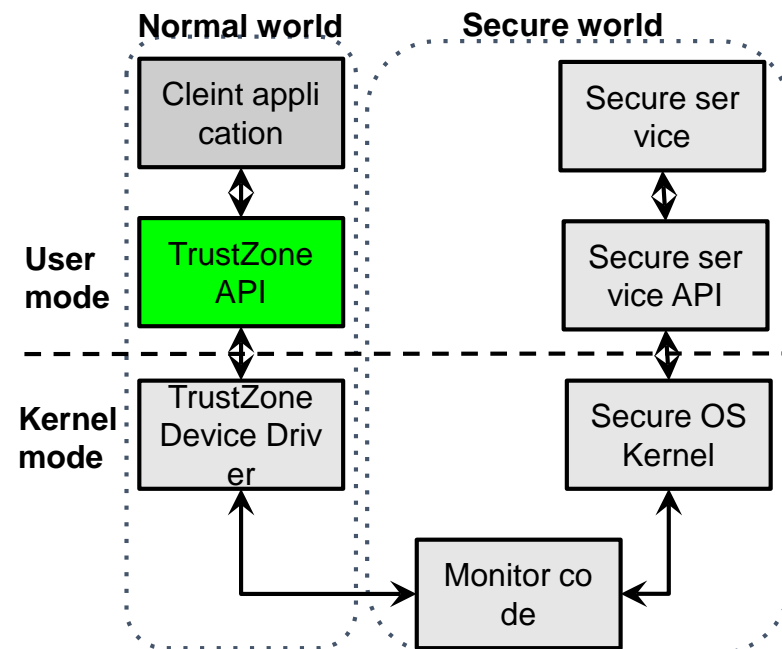
How It Works? (Source Code)

- TrustZone API (User mode in Normal world)
 - Invokes TrustZone kernel driver via ioctl

```
otz_return_t otz_operation_perform(otz_operation_t* ps_operation,  
                                   otz_return_t* pui_service_return )  
{  
    int ret = 0;  
    struct ser_ses_id ses_close;  
    struct ser_ses_id ses_open;  
    struct otz_client_encode_cmd enc;
```

Invoke TrustZone device driver

```
if(ps_operation->type == OTZ_OPERATION_OPEN)  
{  
    ses_open.service_id = ps_operation->session->service_id;  
    ret = ioctl(ps_operation->session->device->fd,  
               OTZ_CLIENT_IOCTL_SES_OPEN_REQ, &ses_open);  
    if(ret < 0){  
        /* Error is detected before reaching the service */  
        *pui_service_return = OTZ_ERROR_GENERIC;  
        ps_operation->ui_state = OTZ_STATE_INVALID;  
        ps_operation->s_errno = errno;  
        /*return Error actually occurred */  
        /*Decoder functions cannot be used on the operation */
```

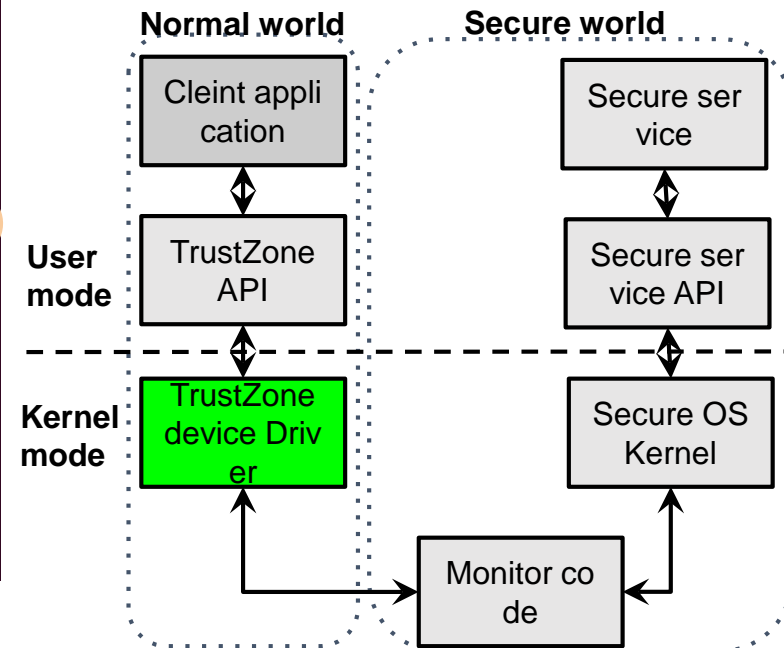


How It Works? (Source Code)

- TrustZone device driver (Kernel in Normal world)
 - Sets params for secure services
 - Invokes SMC instruction to switch to monitor mode

```
static u32 _otz_smc(u32 cmd_addr)
{
    register u32 r0 asm("r0") = CALL_TRUSTZONE_API;
    register u32 r1 asm("r1") = cmd_addr;
    register u32 r2 asm("r2") = OTZ_CMD_TYPE_NS_TO_SECURE;
    do {
        asm volatile(
            __asmeq("%0", "r0")
            __asmeq("%1", "r1")
            __asmeq("%2", "r2")
            "smc    #0    @ switch to secure world\n"
            : "=r" (r0)
            : "r" (r0), "r" (r1), "r" (r2));
    } while (0);
    return r0;
}
```

Invoke SMC to switch the mode



How It Works? (Source Code)

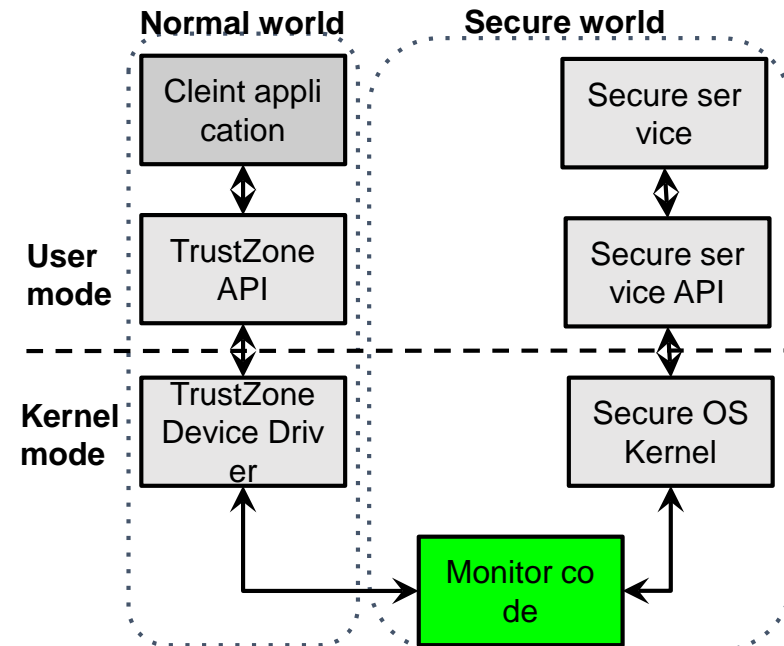
- Monitor code (Monitor mode in Secure world)

```
.align 12
.global monitor
monitor:
@ Monitor's
monitor_reset:
b monitor_reset @ Reset - not used by Monitor
monitor_undef:
b monitor_undef @ Undef - not used by Monitor
b smc_handler @ SMC_Handler
monitor_pref:
b monitor_pref @ Prefetch - can be used by Monitor
monitor_abort:
b monitor_abort @ Data abort - can be used by Monitor
monitor_reserv:
b monitor_reserv @ RESERVE - can be used by Monitor
b monitor_irq @ IRQ - can be used by Monitor
b monitor_fiq_handler @ Monitor FIQ handler

smc_handler:
cmn r0, #0
blt board_smc
CMP r0, #CALL_TRUSTZONE_API
beq tz_api
CMP r0, #CALL_FROM_AMONTZ_STUB
beq tz_api
```

Exception vectors in monitor mode

Exception handler for SMC



How It Works? (Source Code)

- Monitor code (Monitor mode in Secure world)
 - Branches to switch the context to secure world's

```
tz_api:
#ifdef CONFIG_SW_MULTICORE
    push    {r0}
    GET_CPU_ID r0
    cmp     r0, #0
    pop     {r0}
    beq     1f
    b       smc_error
#endif

#ifdef CONFIG_SW_DEDICATED_TEE
1:  push    {r4, lr} /* the corresponding pops happens from
                    save_context */

    push    {r0 - r3}

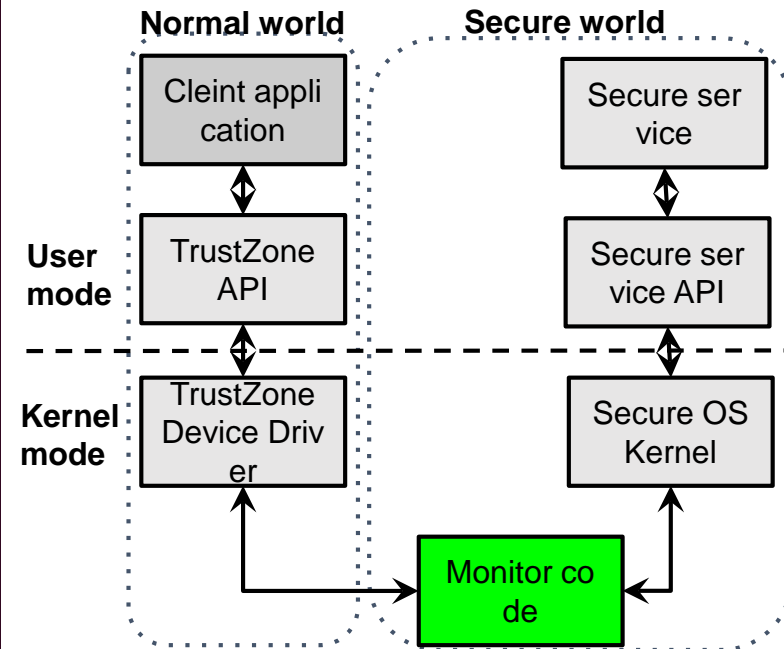
    /* Copy args to params stack */
    ldr     r4, =params_stack
    stmia   r4, {r0-r3}

    ldr     r1, =valid_params_flag
    mov     r2, #0x1
    str     r2, [r1]

    b       mon_switchto_sworld

```

Switch the context to secure world's



How It Works? (Source Code)

- Monitor code (Monitor mode in Secure world)
 - Clear NS-bit and save the context of normal world

```
.func mon_switchto_sworld
mon_switchto_sworld:

@Move to Secure
scr_nsbit_clear r0

GET_CORE_CONTEXT ns_sys_current
bl save_context
GET_CORE_CONTEXT s_sys_current
bl restore_context
@ Clear local monitor
@ -----
clrex

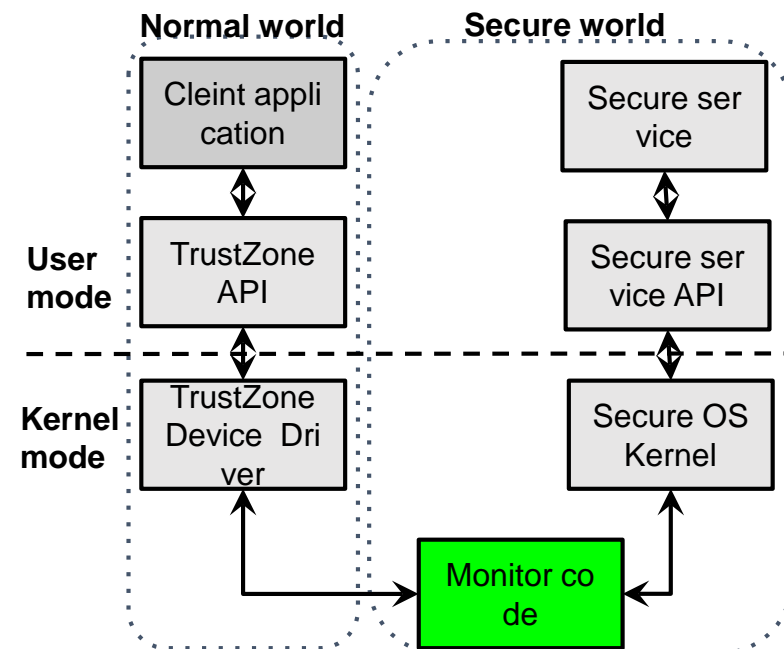
movs    pc, lr
.endfunc

.func save_context
.global save_context
save_context:
@ Save general purpose registers, CPSR and LR
@ -----
mov     r4, r0
pop     {r0 - r3}
stmia   r4!, {r0 - r3}
mov     r0, r4

@ Retrieve from stack
@ r0 - r3
```

Exception vectors in monitor mode

Save the normal world's context



How It Works? (Source Code)

- Monitor code (Monitor mode in Secure world)
 - Restore the context of secure world
 - General registers, SPSR, LR, PC ...

```
.func restore_context
.global restore_context
restore_context:
@ Restore other world's registers, SPSR and LR
@ -----
push    {lr}

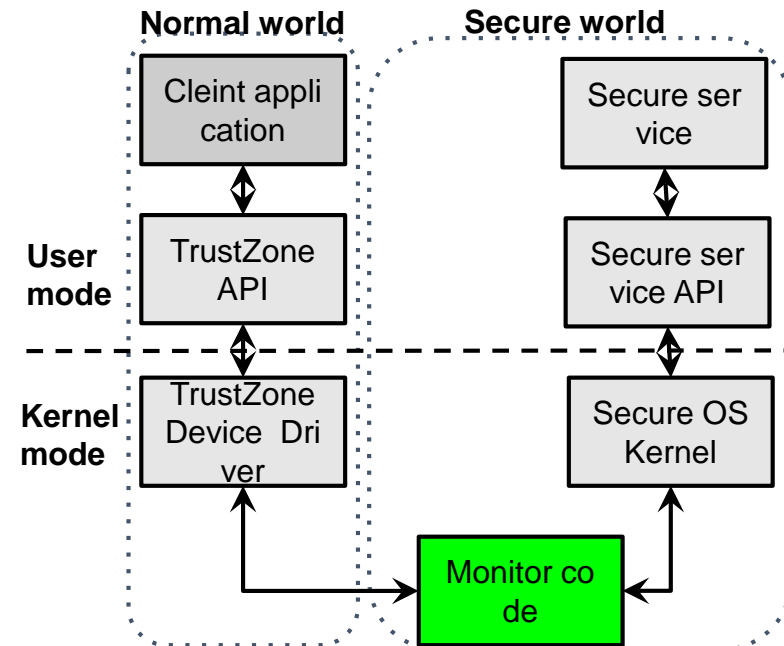
mov r4, r0
ldmia r4!, {r0 - r3}

push {r0, r1}
mov r0, r4

ldmia r0!, {r4 - r12}           @r4 - r12
ldmia r0!, {r1, lr}             @spsr, lr
msr spsr_cxsf, r1

@Restore banked registers
cps #Mode_SVC
ldmia r0!, {r1, r13, lr}
msr spsr_cxsf, r1
```

Restore the secure world's context



How It Works? (Source Code)

- Dispatcher service (User mode in Secure world)
 - Parses requests and dispatches secure tasks

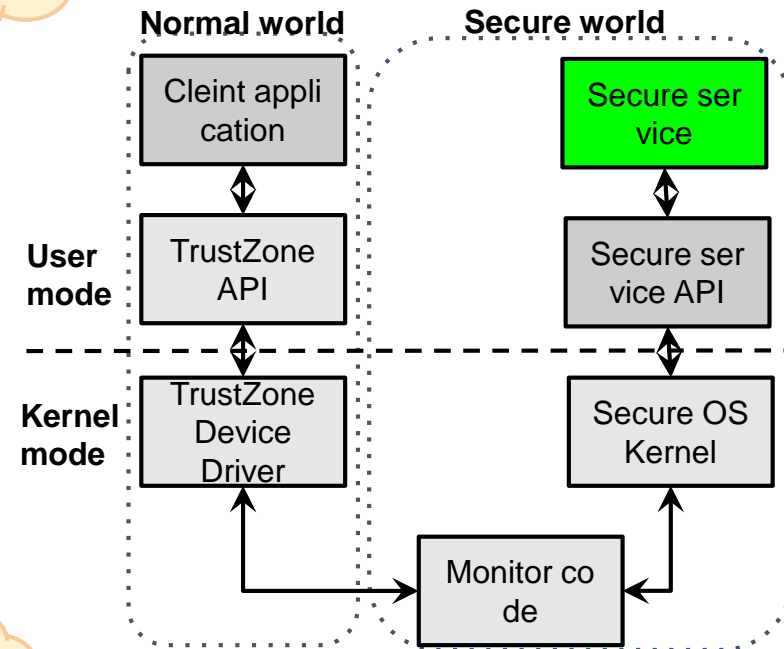
```
void dispatch_task(int cur_task_id)
{
    int ret_val;
    int svc_id, task_id, cmd_id, cmd_type;

    (cmd_id == OTZ_GLOBAL_CMD_ID_OPEN_SESSION)) {
    if(cmd_type == OTZ_CMD_TYPE_NS_TO_SECURE) {
        /* sw_buddy_print_state(); */
        ret_val = open_session_from_ns((void*)params_stack[1]);
        if(ret_val == OTZ_ENOMEM)
            set_secure_api_ret(SMC_ENOMEM);

    } else if(svc_id != OTZ_SVC_INVALID &&
               task_id != 0 &&
               cmd_id != OTZ_GLOBAL_CMD_ID_INVALID) {
        start_task(task_id, params_stack);
    } else {
        sw_printf("SW: Invalid service id %x\n", svc_id);
        set_secure_api_ret(0);
    }
}
schedule();
```

-Parses a request
-Create a task

Start a secure task
to handle the request



How It Works? (Source Code)

- Secure service API (User mode in Secure world)
 - Invokes system calls (memory maps, scheduling...)

```
int open_session_from_ns(void *param)
{
    int ret_val;
    int *svc_id = NULL;
    sa_config_t sa_config;
    pa_t cmd_phy;
    struct otz_smc_cmd *cmd = NULL;
    void *session_context = NULL;

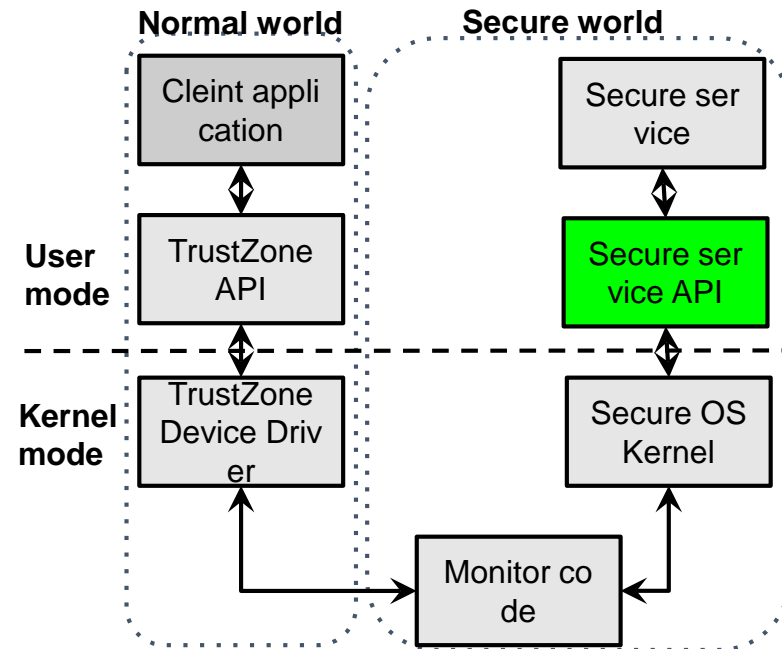
    cmd_phy = (pa_t) param;

    if(map_to_ns(cmd_phy, (va_t*) &cmd)) {
        ret_val = OTZ_ENOMEM;
        goto ret_func;
    }

    void schedule(void)
    {
#ifdef CONFIG_EMULATE_FIQ
        emulate_timer_irq();
#else
        asm volatile("swi #0xbbbb");
#endif
    }
}
```

System calls for kernel services

System calls for kernel services



How It Works? (Source Code)

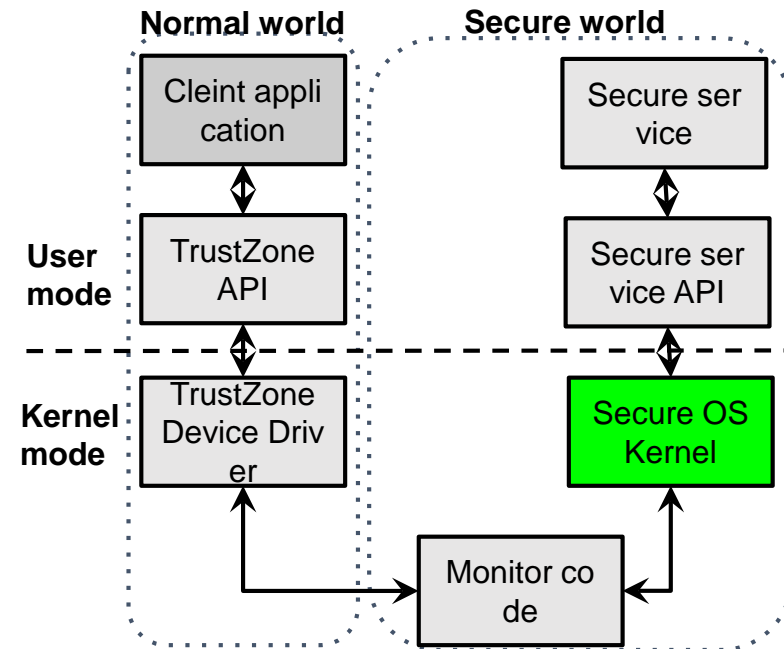
- Secure OS (Kernel mode in Secure world)

```
.global secure_exception_vectors
secure_exception_vectors:
/*
Vector table
*/
B    _reset_handler
B    _undefined_handler
B    _swi_handler
B    _prefetch_handler
B    _abort_handler
b    _reserved_vector
B    _irq_handler
B    _fiq_handler
```

Exception vector for kernel mode in Secure world

```
.global _swi_handler
.func _swi_handler
_swi_handler:
    stmfd    sp!, {r0-r12, lr}
    ldr      r0, [lr, #-4]
    bic      r0, r0, #0xff000000
    mrs      r2, spsr
    stmfd    sp!, {r2}
    mov      r1, sp
    bl       swi_handler
    ldmfd    sp!, {r2}
    msr      spsr, r2
    push     {r0}
    mov      r0, #0
    dsb
    pop      {r0}
    ldmfd    sp!, {r0-r12, pc}^
.endfunc
```

System-call exception handler



How It Works? (Source Code)

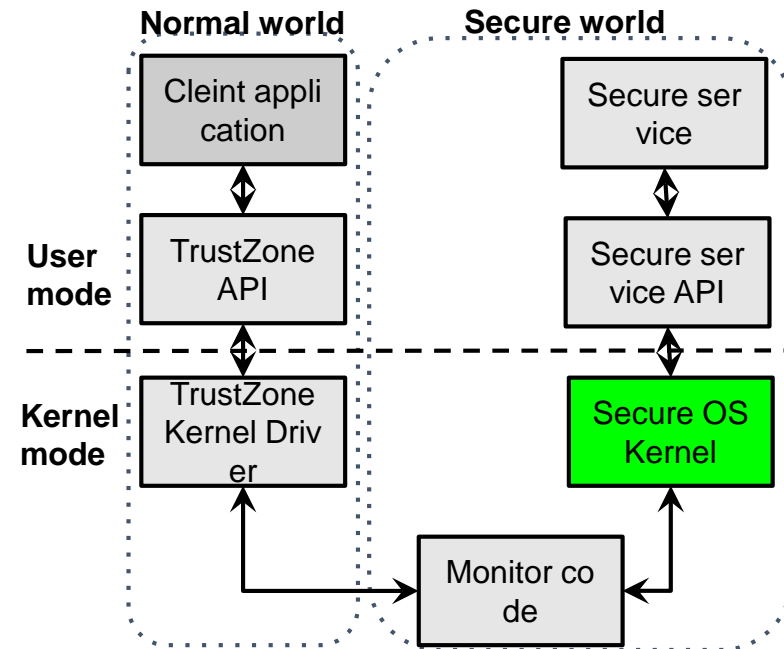
- Secure OS (Kernel mode in Secure world)
 - System call handler (triggers a scheduler)

```
void swi_handler(int swi_id, struct swi_temp_regs *regs)
{
    /*    int i = 0; */

#ifdef CONFIG_KSPACE_PROTECTION
    int ret=0;
    struct list *l1;
    struct sw_file_operations *dev_info;
#endif

    switch(swi_id) {
        case 0xbbbb:
        case 0xcccc:
            /*
             * for(i = 0; i < 14; i++)
             *     tz_printf("swi temp regs[%x] = 0x%x\n", i, regs[i]);
             */
            temp swi regs = regs;
            scheduler();
            break;
    }
}
```

Triggers a Secure world's scheduler



How It Works? (Source Code)

- Secure OS (Kernel mode in Secure world)
 - Scheduler: new task scheduling

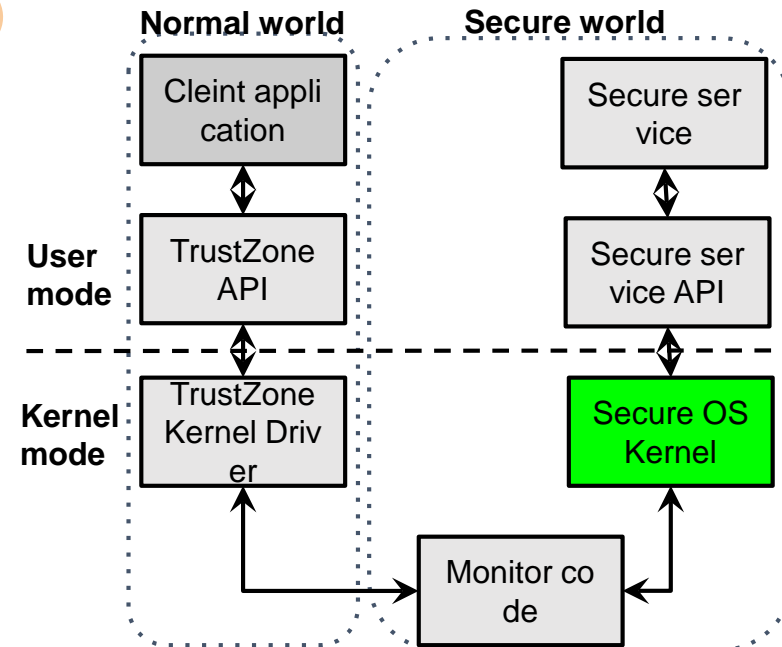
```
void scheduler(void)
{
    struct sw_task *next_task;
    struct sw_task *current_task;

    current_task = get_current_task();
    next_task = get_next_task();

    if(current_task != next_task) {
        if(next_task) {
            next_task->state = TASK_STATE_RUNNING;
            update_current_task(next_task);
        }
        task_context_switch(next_task, current_task);
    }

    if(!next_task)
        disable_timer();
}
```

Context switch for a new task



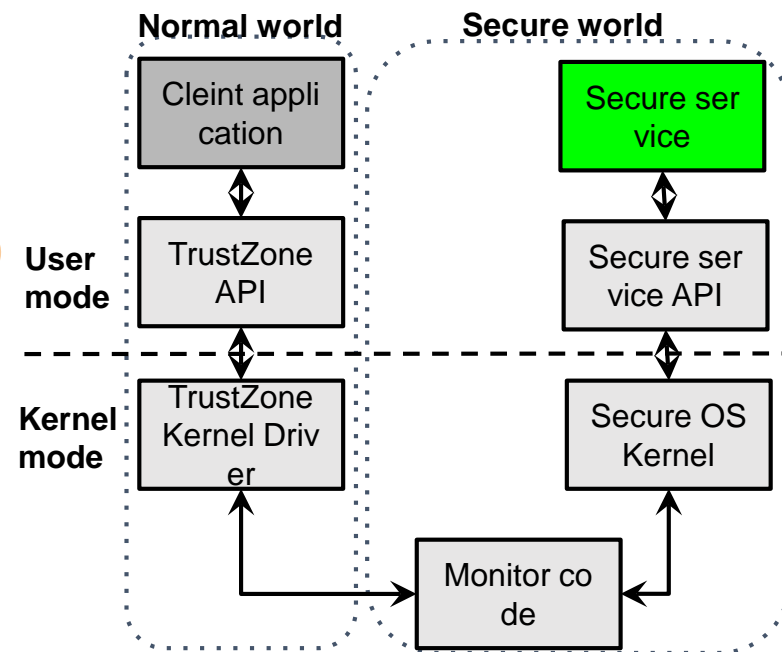
How It Works? (Source Code)

- Crypto service (User mode in Secure world)
 - Handle the actual request (message decryption)

```
void crypto_task(int task_id, sw_tls* tls)
{
    tls->ret_val = process_otzapi(task_id, tls);
    handle_task_return(task_id, tls);
    while(1);
}
```

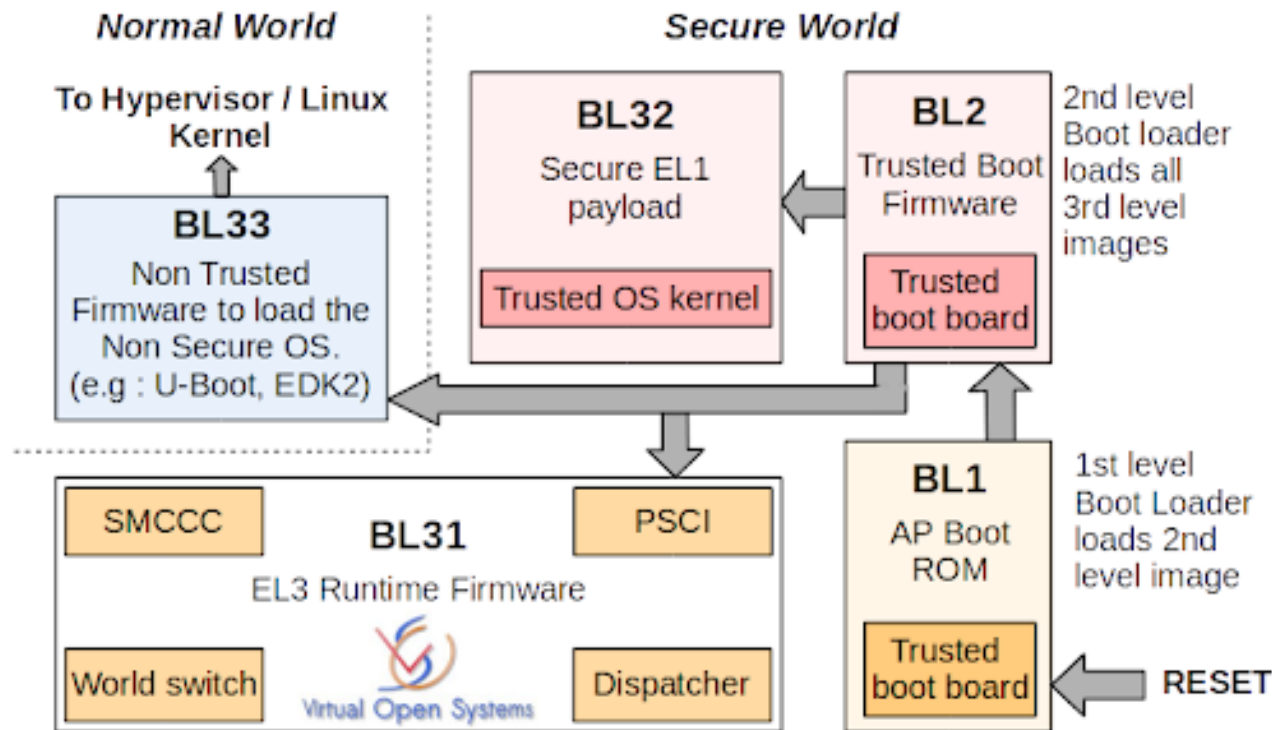
```
break;
case OTZ_CRYPT_CMD_ID_CIPHER_AES_128_CBC:
case OTZ_CRYPT_CMD_ID_CIPHER_AES_128_ECB:
case OTZ_CRYPT_CMD_ID_CIPHER_AES_128_CTR:
case OTZ_CRYPT_CMD_ID_CIPHER_AES_128_XTS:
case OTZ_CRYPT_CMD_ID_CIPHER_DES_ECB:
case OTZ_CRYPT_CMD_ID_CIPHER_DES_CBC:
case OTZ_CRYPT_CMD_ID_CIPHER_DES3_ECB:
case OTZ_CRYPT_CMD_ID_CIPHER_DES3_CBC:
    ret_val = process_otz_crypto_cipher_cmd(req_buf, req_buf_len,
        resp_buf, res_buf_len, meta_data,
        ret_res_buf_len, svc_cmd_id);
    break;
default:
    ret_val = SMC_EOPNOTSUPP;
    break;
```

Decrypt the message from Normal world



ARM Trusted Firmware

ARM Trusted Firmware



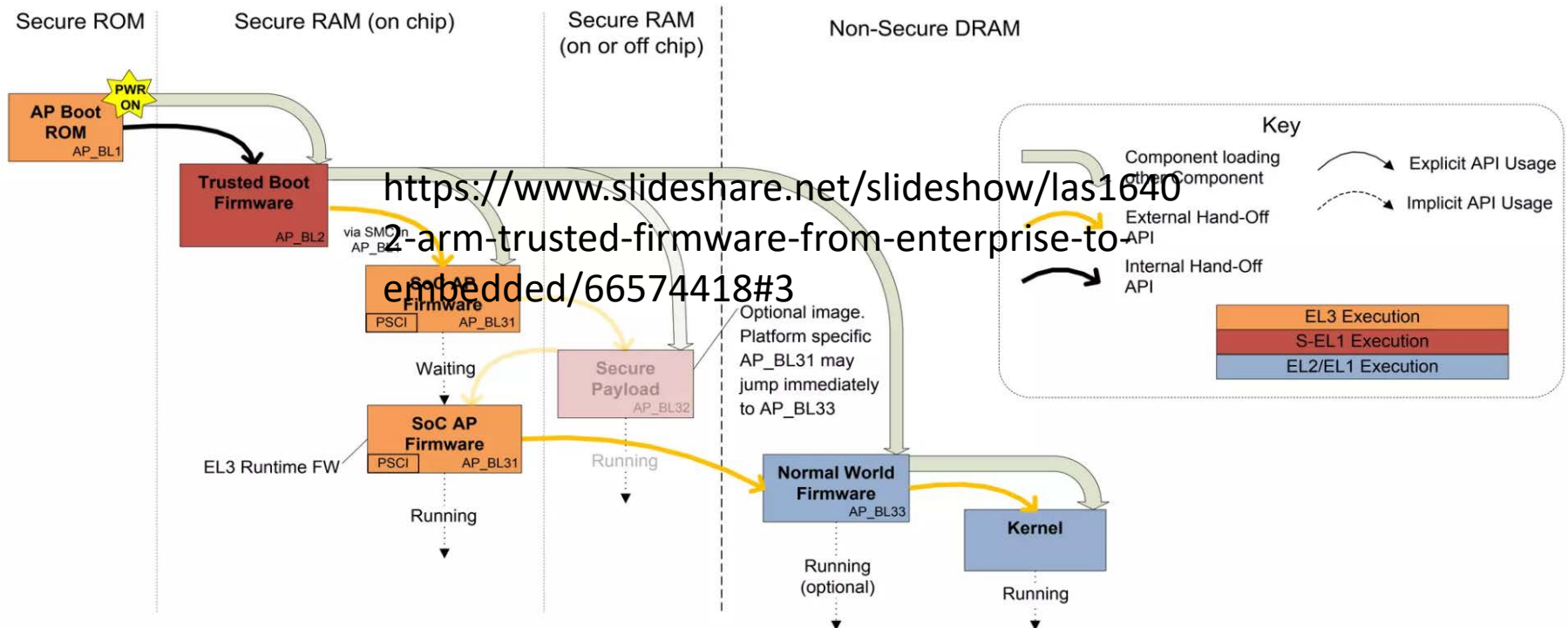
Key

EL3 Execution
Secure EL1 Execution
Normal EL2/EL1 Execution

Glossary

EDK2 – EFI Development Kit 2
EL – Exception Level
PSCI – Power State Control Interface
BL – Boot Loader
SMC – Secure Monitor Call

Booting Process on AArch64

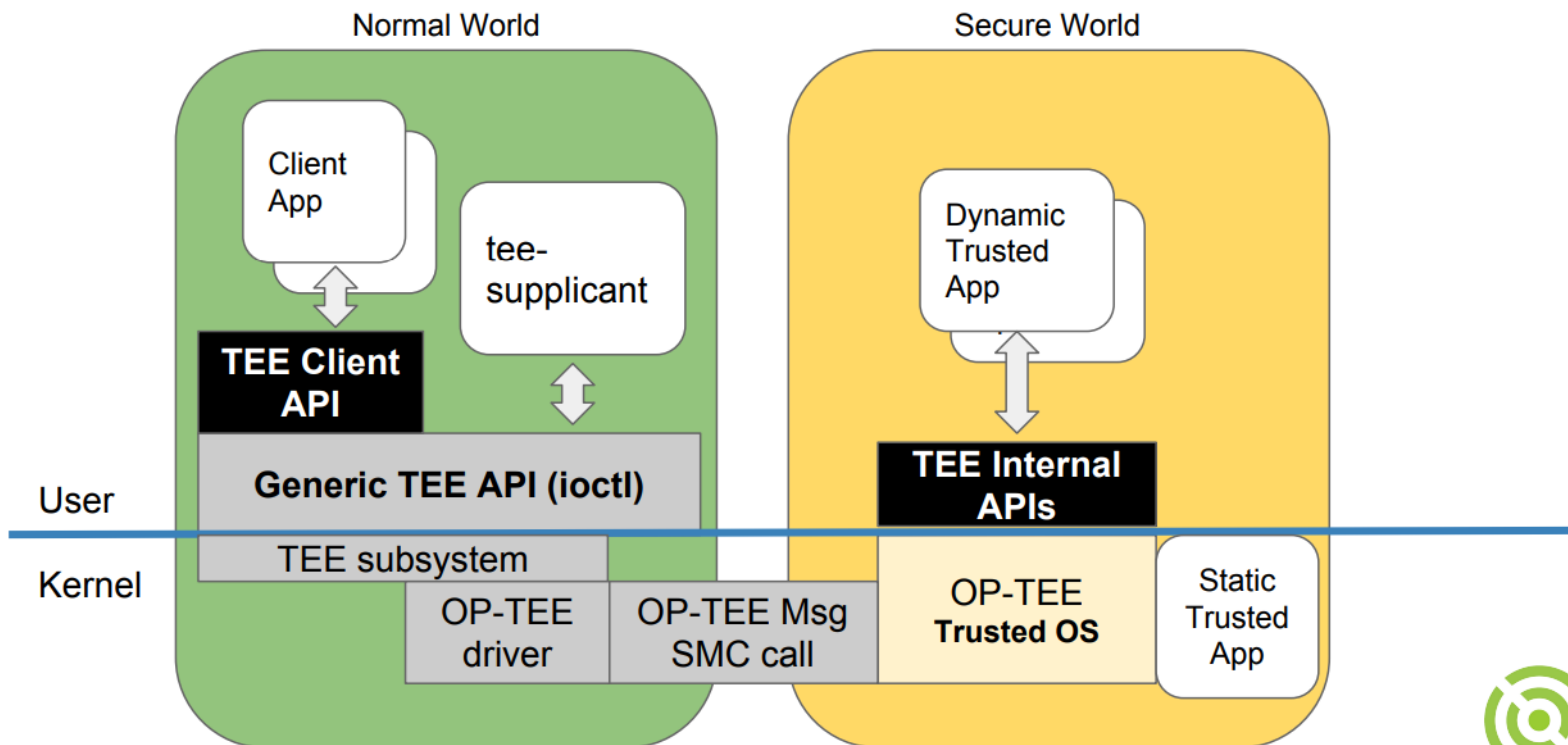


<https://www.slideshare.net/slideshow/las1640-2-arm-trusted-firmware-from-enterprise-to-embedded/66574418#3>



OP-TEE

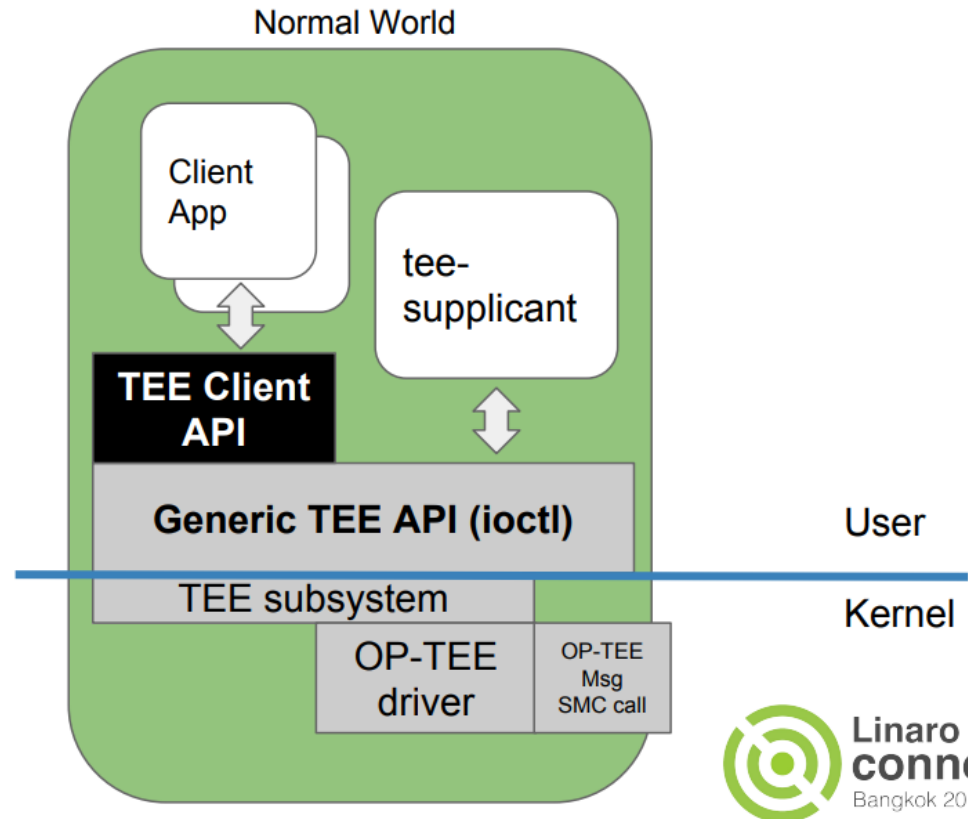
OP-TEE Architecture



OP-TEE: Linux Kernel Subsystem

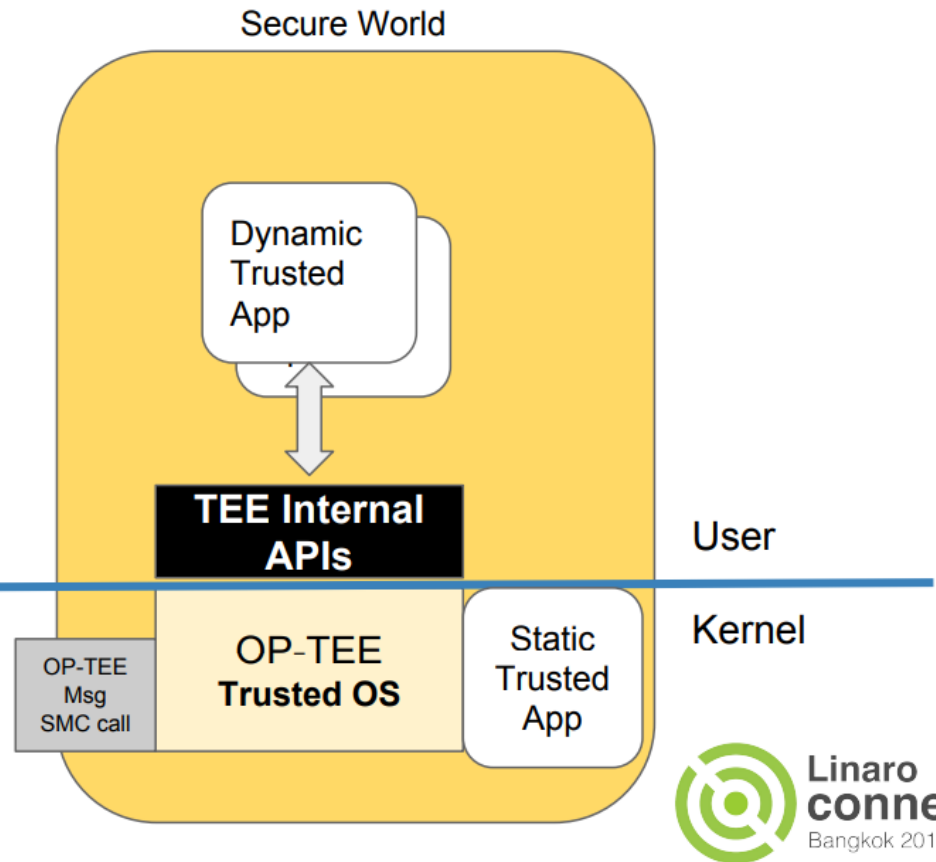
- **TEE subsystem:**
 - Manages Shared Memory,
 - Provides generic API as ioctl.
- **tee-supplciant:**
 - Helper process for TEE.
- **OP-TEE driver:**
 - Forwards command from the Clients to OP-TEE,
 - Manages RPC requests from OP-TEE to the supplicant.

For more details, please read [optee_design.md](#) at GitHub.



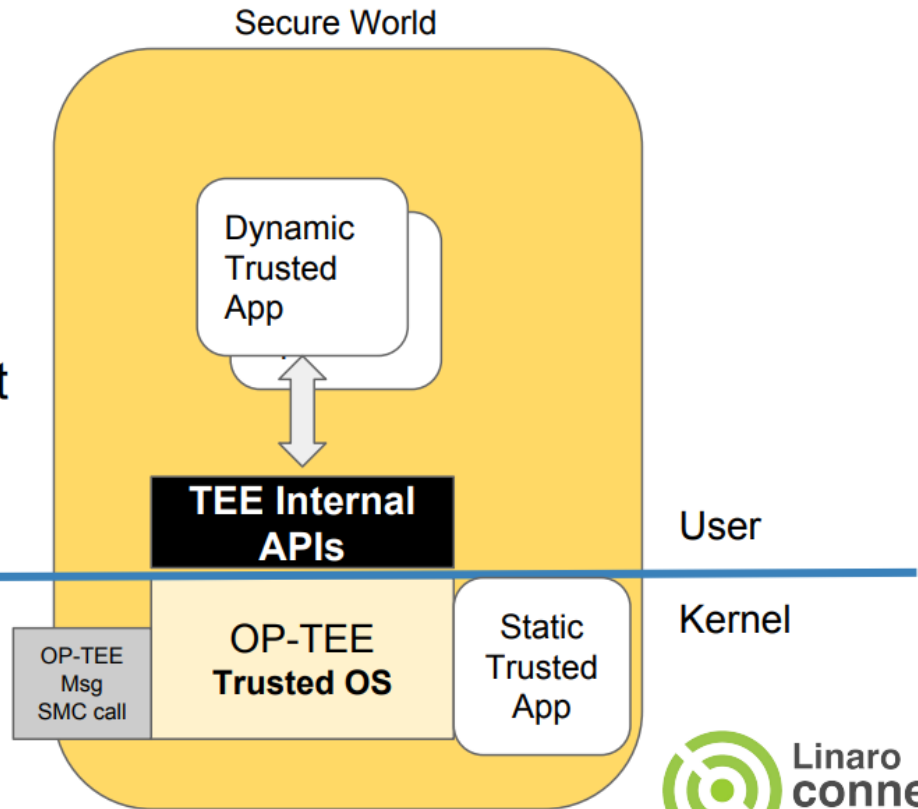
OP-TEE: Managing Trusted Apps

- Trusted Apps:
 - **Static:** run in kernel mode,
 - **Dynamic:** run in user mode:
 - Stored in File System,
 - Loaded by OP-TEE
 - Using tee_suppllicant.
- MMU L1 Translation tables:
 - Large: 4GiB for OP-TEE kernel,
 - Small tables: 32MiB:
 - TA Virtual Memory,
 - One per thread.



OP-TEE: Comm. & Scheduling

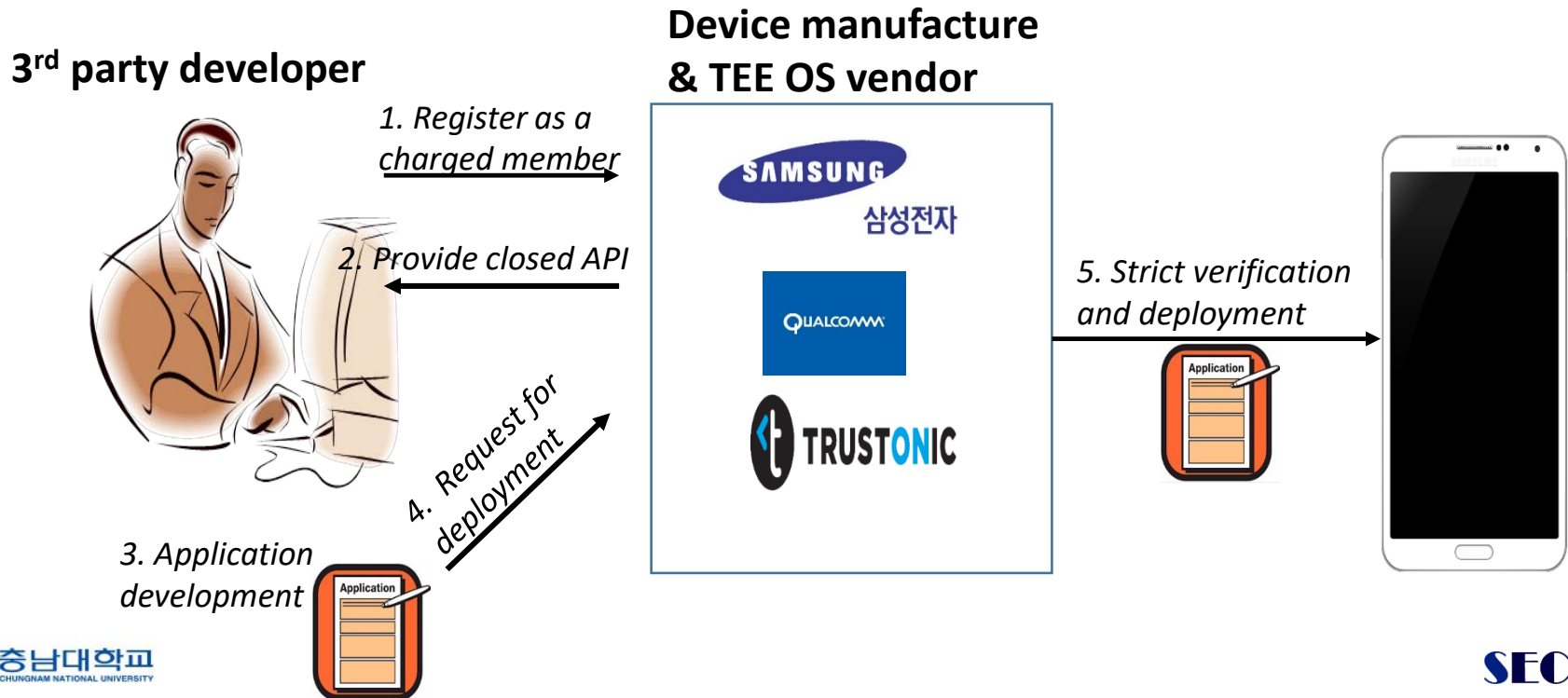
- Entry into secure world from SMC (or FIQ arriving).
- Command arriving → Allocated to thread (if available), TA context set up and called;
 - If supplicant needed, RPC is started and thread suspended.
- Return to normal world on task completion, RPC (or IRQ arriving).



Use cases

Closed Development Model

- ❖ Charged service, sometimes expensive
- ❖ TEE is maintained by TEE service providers
 - Device manufacturer: Samsung, Qualcomm
 - TEE OS provider: Trustonic



Open Source TEE

❖ Google Trusty

- <https://source.android.com/security/trusty/>
- Consist of TEE OS, TrustZone driver and APIs

❖ Linaro OP-TEE

- <https://wiki.linaro.org/WorkingGroups/Security/OP-TEE>
- Support various platforms

Table with 3 columns: Platform, Composite PLATFORM flag, and Publicly available?

Platform	Composite PLATFORM flag	Publicly available?
Allwinner A80 Board	PLATFORM=sunxi	No
ARM Juno Board	PLATFORM=vexpress-juno	Yes
FSL ls1021a	PLATFORM=ls-ls1021atwr	Yes
FSL i.MX6 Quad SABRE Lite Board	PLATFORM=imx-mx6qsabrelite	Yes
FSL i.MX6 Quad SABRE SD Board	PLATFORM=imx-mx6qsabresd	Yes
FSL i.MX6 UltraLite EVK Board	PLATFORM=imx-mx6ulevk	Yes
NXP i.MX7Dual SabreSD Board	PLATFORM=imx-mx7dsabresd	Yes
ARM Foundation FVP	PLATFORM=vexpress-fvp	Yes
HiSilicon D02	PLATFORM=d02	No
HiKey Board (HiSilicon Kirin 620)	PLATFORM=hikey or PLATFORM=hikey-hikey	Yes
HiKey960 Board (HiSilicon Kirin 960)	PLATFORM=hikey-hikey960	Yes
MediaTek MT8173 EVB Board	PLATFORM=mediatek-mt8173	No
QEMU	PLATFORM=vexpress-qemu_virt	Yes
QEMUv8	PLATFORM=vexpress-qemu_armv8a	Yes
Raspberry Pi 3	PLATFORM=rpi3	Yes
Renesas RCAR	PLATFORM=rcar	No

TrustZone with Different Arch.

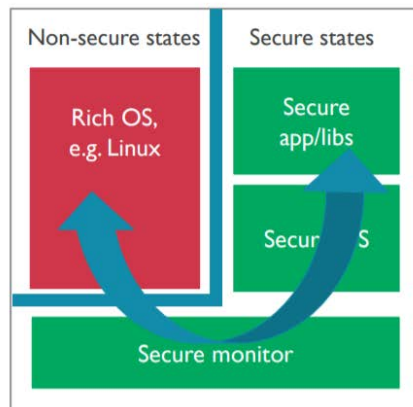
❖ ARMv7

- 32-bit architecture

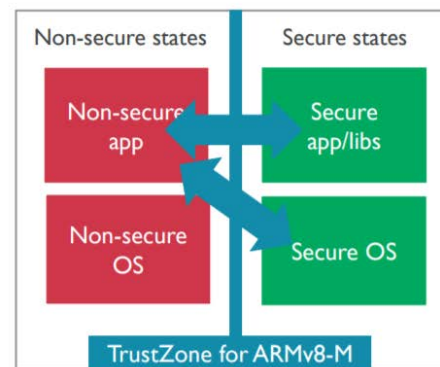
❖ ARMv8

- Both 32-bit and 64-bit architectures
- ARMv8-A,R: for high-end devices (e.g., mobile phone)
- ARMv8-M: for microcontrollers

TrustZone for ARMv8-A



TrustZone for ARMv8-M



Secure transitions handled by the processor to maintain embedded class latency

Figure retrieved from:
http://www2.keil.com/docs/default-source/default-document-library/using_trustzone_on_arm_cortex-m23_and_cortex-m33.pdf?sfvrsn=2

TrustZone in Industry

❖ Samsung

- KNOX: security platform
- Samsung pay: TEE service

❖ Qualcomm

- QSEE : Qualcomm TEE

❖ Trustonic

- Trustonic Secured Platforms (TSP)

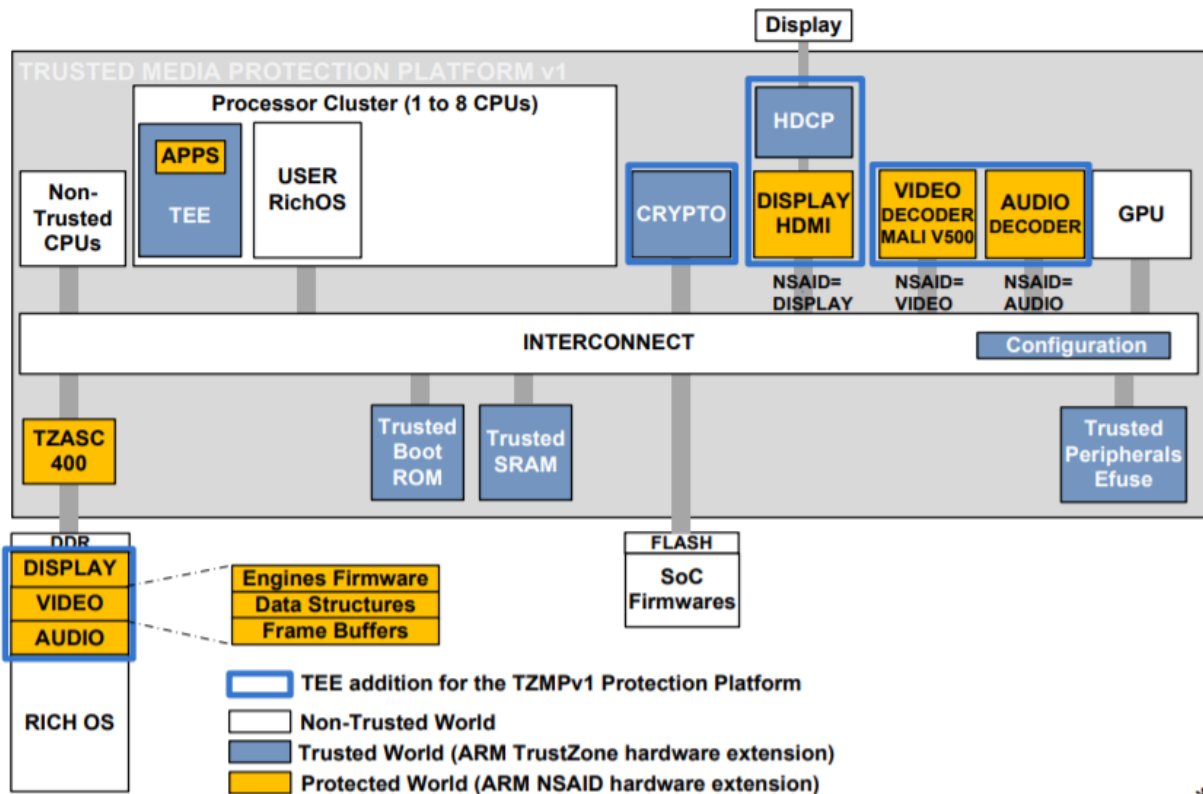
❖ Google

- Trusty: open sourced TEE

❖ ARM

- Mbed

Content Protection



DRM & Keys protected by TEE
Hardware isolated video path



*Non-Secure Access ID

ARM®

MDM & Enterprise

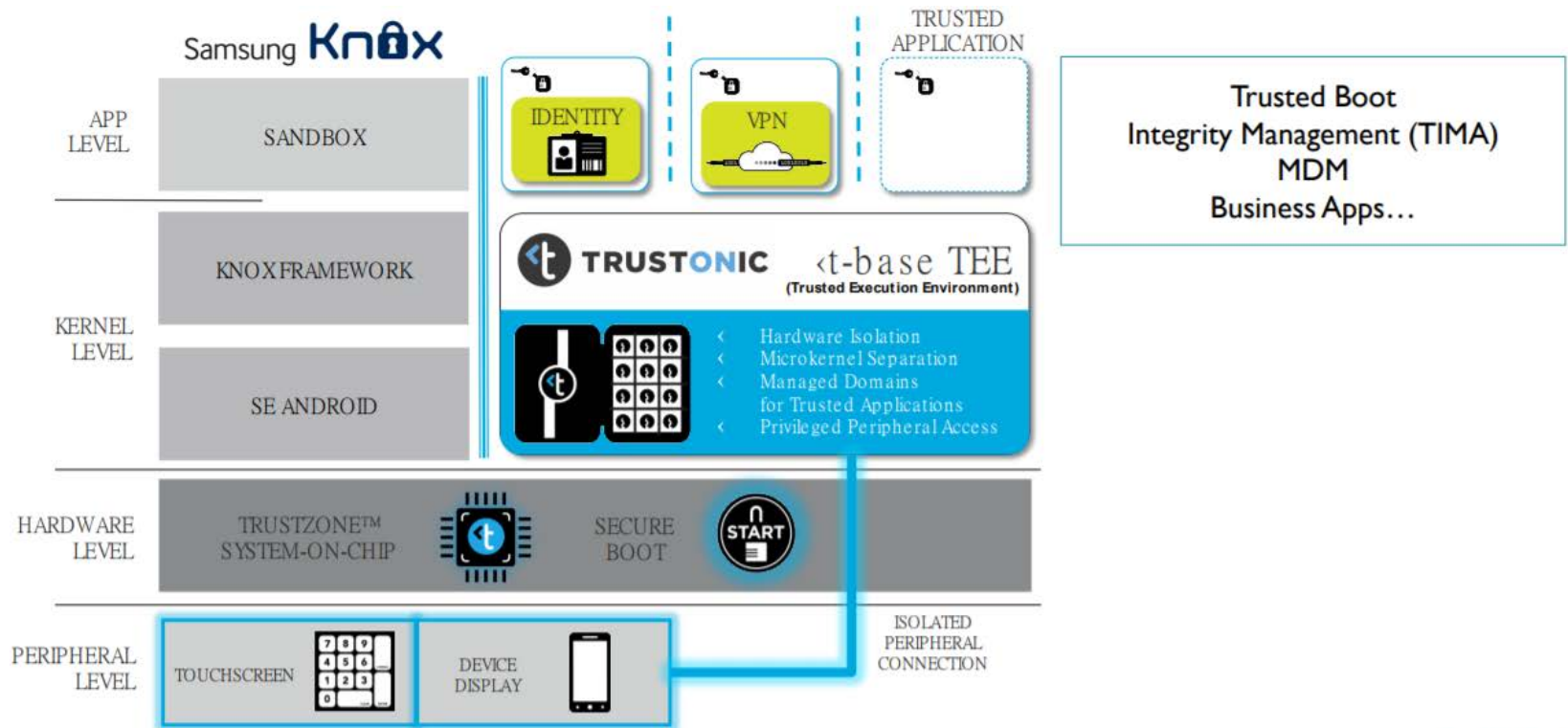


Figure retrieved from: http://www.armtechforum.com.cn/2014/sz/A-8_FIDOandTEE-

SimplerStrongerAuthentication.pdf

Secure Payment

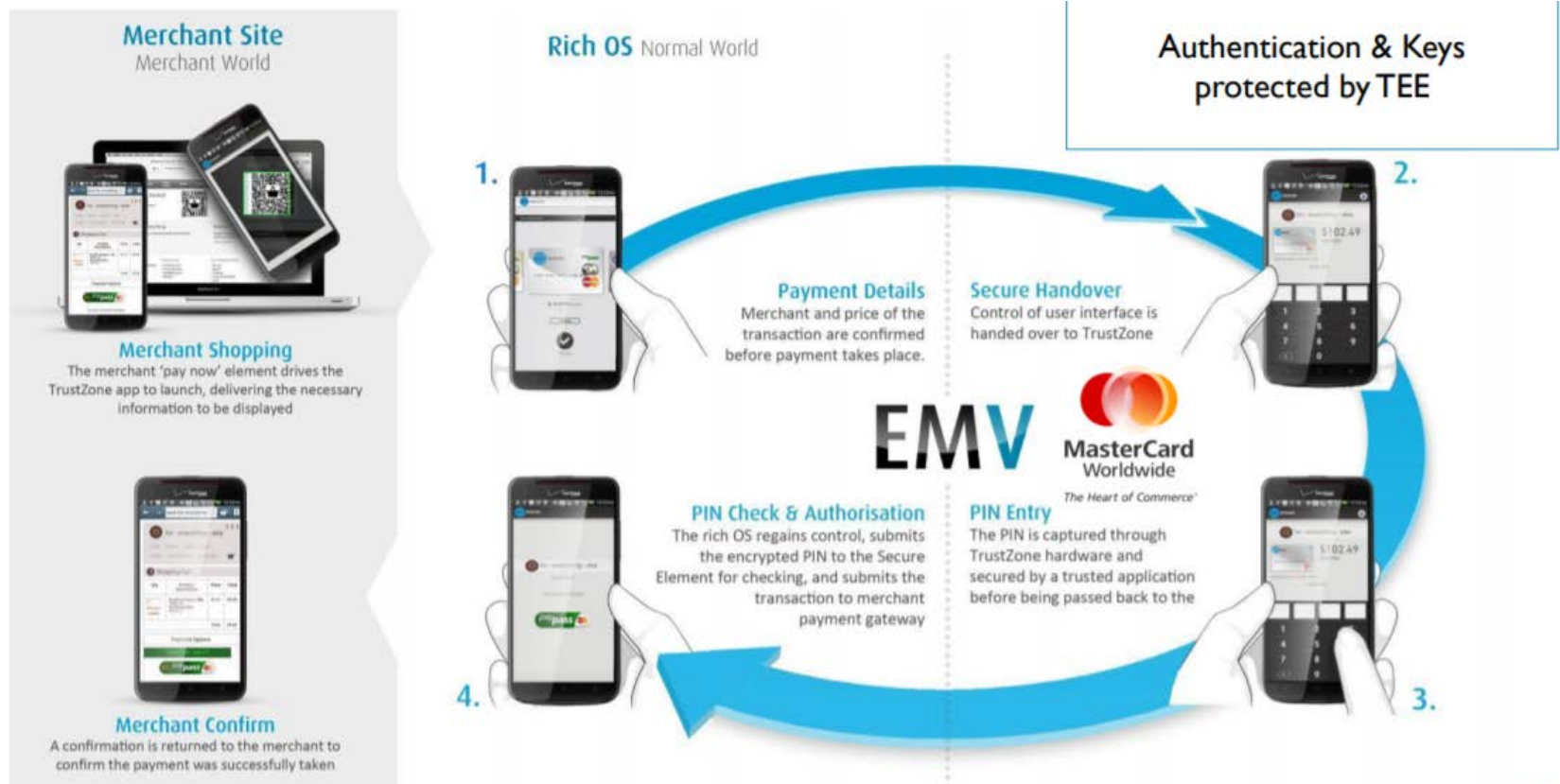


Figure retrieved from: http://www.armtechforum.com.cn/2014/sz/A-8_FIDOandTEE-SimplerStrongerAuthentication.pdf

Privacy-preserving Contact Tracing

Tracing without violating privacy

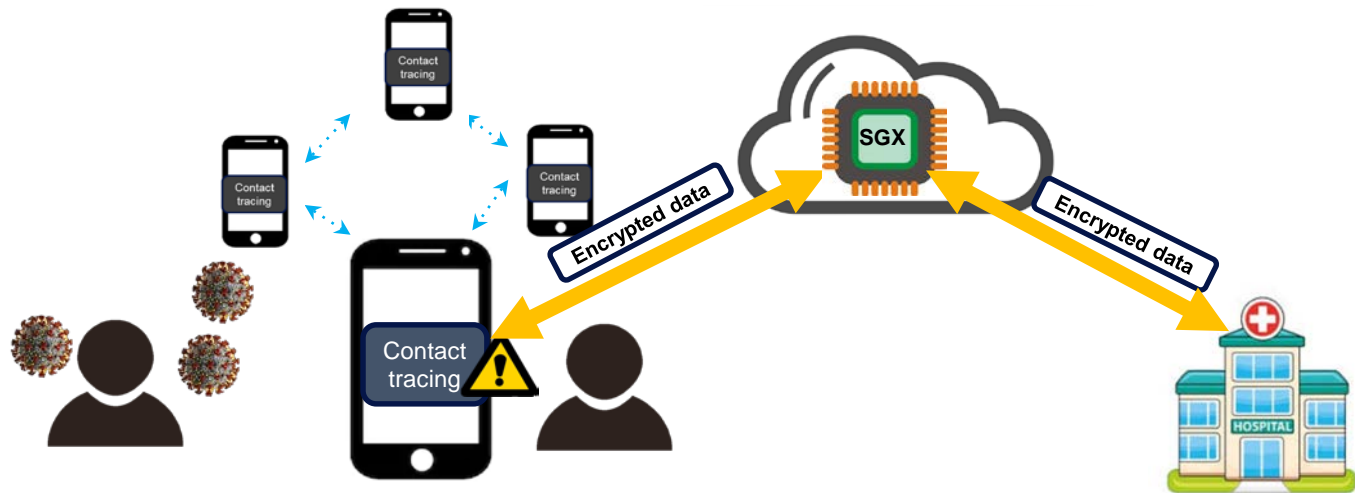


Figure from: Towards Confidential Computing with Trusted Execution Environment

Medical Data Sharing

Data sharing without violating privacy

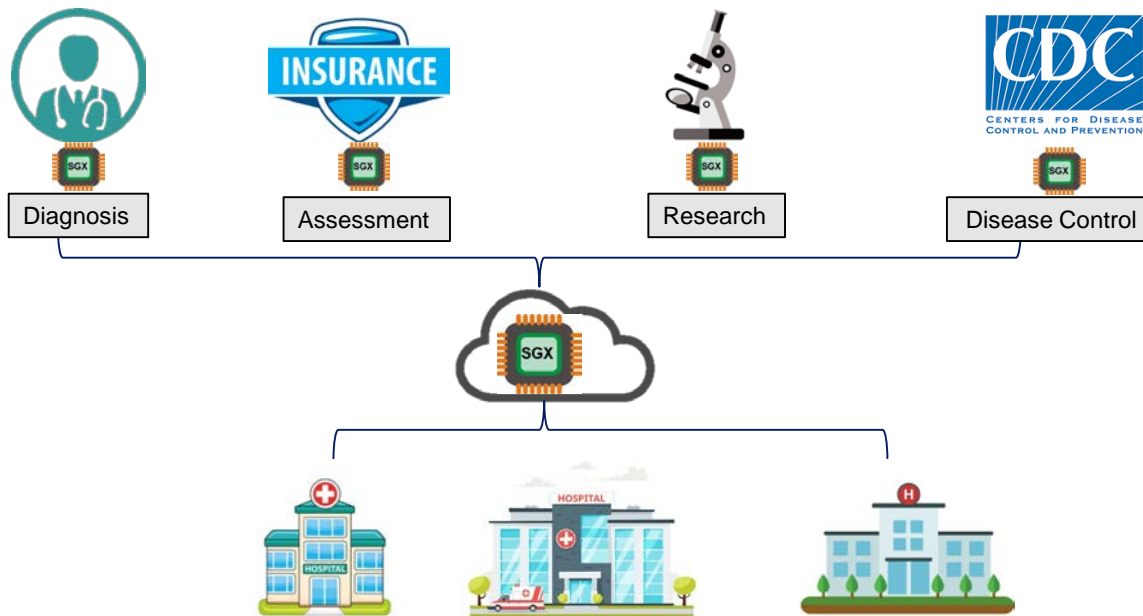


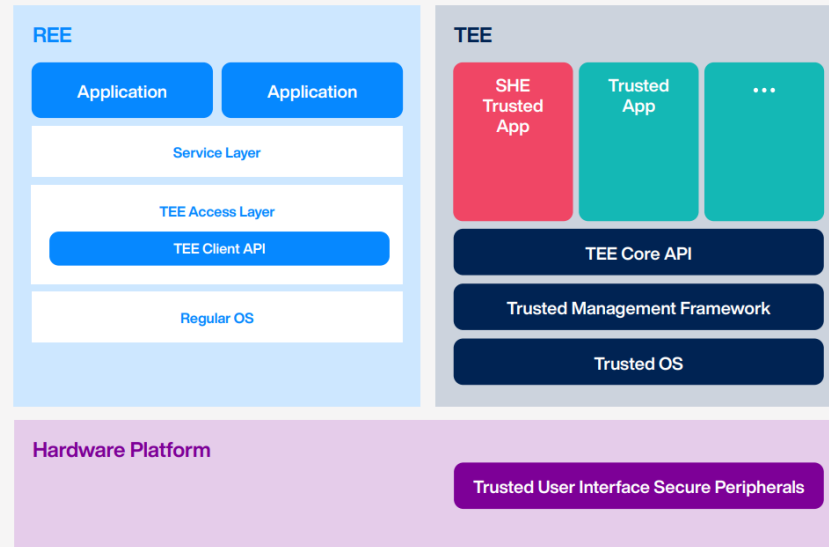
Figure from: Towards Confidential Computing with Trusted Execution Environment

Automotive Systems Security



4.3.1 Trusted Execution Environment Architecture

Figure 16:
GlobalPlatform Trusted Execution Environment Architecture



Drone Security



SiMa^{ai}



APPLICATIONS

High-end consumer
Professional enterprise
Logistics
Construction and public safety
Agriculture and mining
Utilities and building inspection



- + Authentication and encryption capabilities enable you to protect your sensitive data and IP.
- + Integrated secure boot, Arm TrustZone, secure communication, and data security.

Satellite Security



Jamie Oglethorpe  · Follow

BSc (Hons) in Computer Science, Loughborough University (Graduated 1976) · 1y

Starlink terminal hacked!



The Hacking of Starlink Terminals Has Begun

It cost a researcher only \$25 worth of parts to create a tool that allows...

🔗 <https://www.wired.com/story/starlink-internet-dish-hack/>



U-Boot 2020.04-gddb7afb (Apr 16 2021 - 21:10:45 +0000)

```
Model: Catson
DRAM: 1004 MiB
MMC:   Fast boot:eMMC: 8xbit - div2
stm-sdhci: 0
Trn: nulldev
Out: serial
Err: serial
CPU ID: 0x00020100 0x87080245 0xb9cadb91
Detected Board rev: #rev2_prot2
sdhci_set_clock: Timeout to wait cmd & data inhibit
FPGA: 3 FIP2: 3
BOOT SLOT 0
Net: Net Initialization Skipped
No ethernet found.
```

Board: SPACEX CATSON UTERM

```
=====
= Type 'falcon' to stop boot process =
=====
```

Continuing through the boot process we can see that U-Boot loads a kernel, ramdisk and flattened Device Tree (FDT) from a Flattened uImage Tree (FIT) image that is stored on an embedded MultiMediaCard (eMMC).

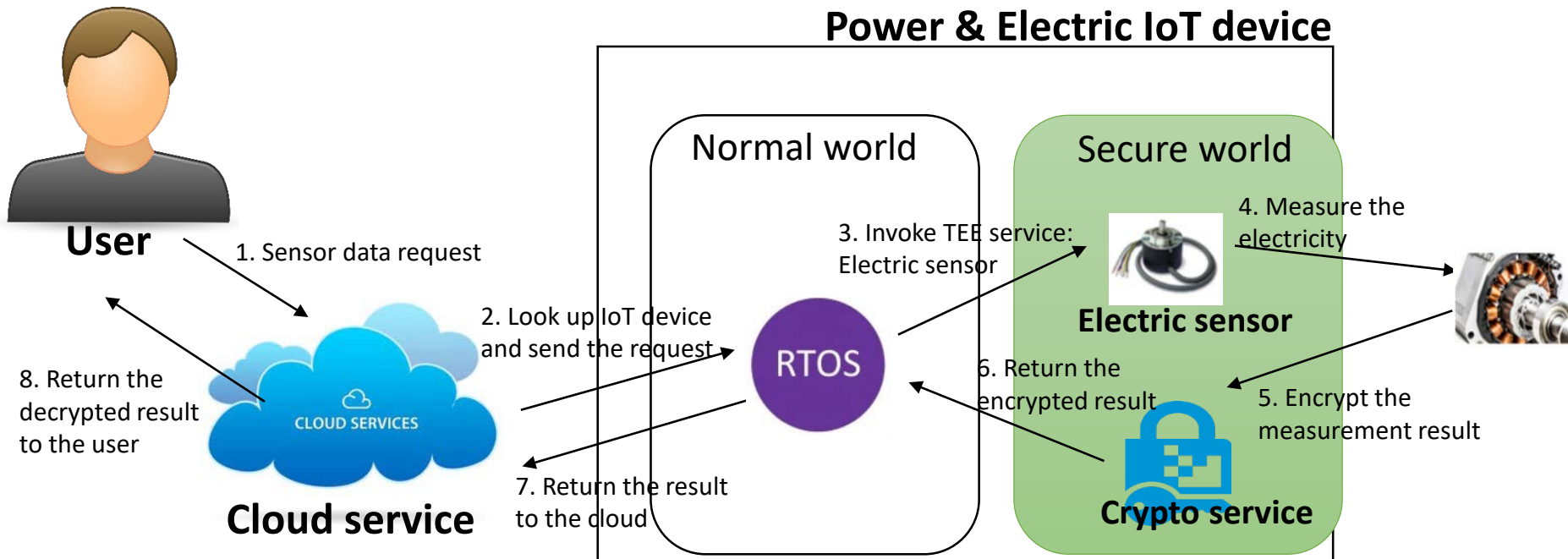
We can also see that the integrity (SHA256) and authenticity (RSA 2048) of the kernel, ramdisk and FDT is being checked. While we would have to perform some more tests it appears that a full trusted boot chain (TF-A) is implemented from the early stage ROM bootloader all the way down to the Linux operating system.

```
switch to partitions #0, OK
mmc0(part 0) is current device
```

```
MMC read: dev # 0, block # 98304, count 49152 ... 49152 blocks read:
## Loading kernel from FIT Image at a2000000 ...
Using 'rev2_proto2@1' configuration
Verifying Hash Integrity ... sha256,rsa2048:dev+ OK
Trying 'kernel@1' kernel subimage
Description: compressed kernel
Created: 2021-04-16 21:10:45 UTC
```

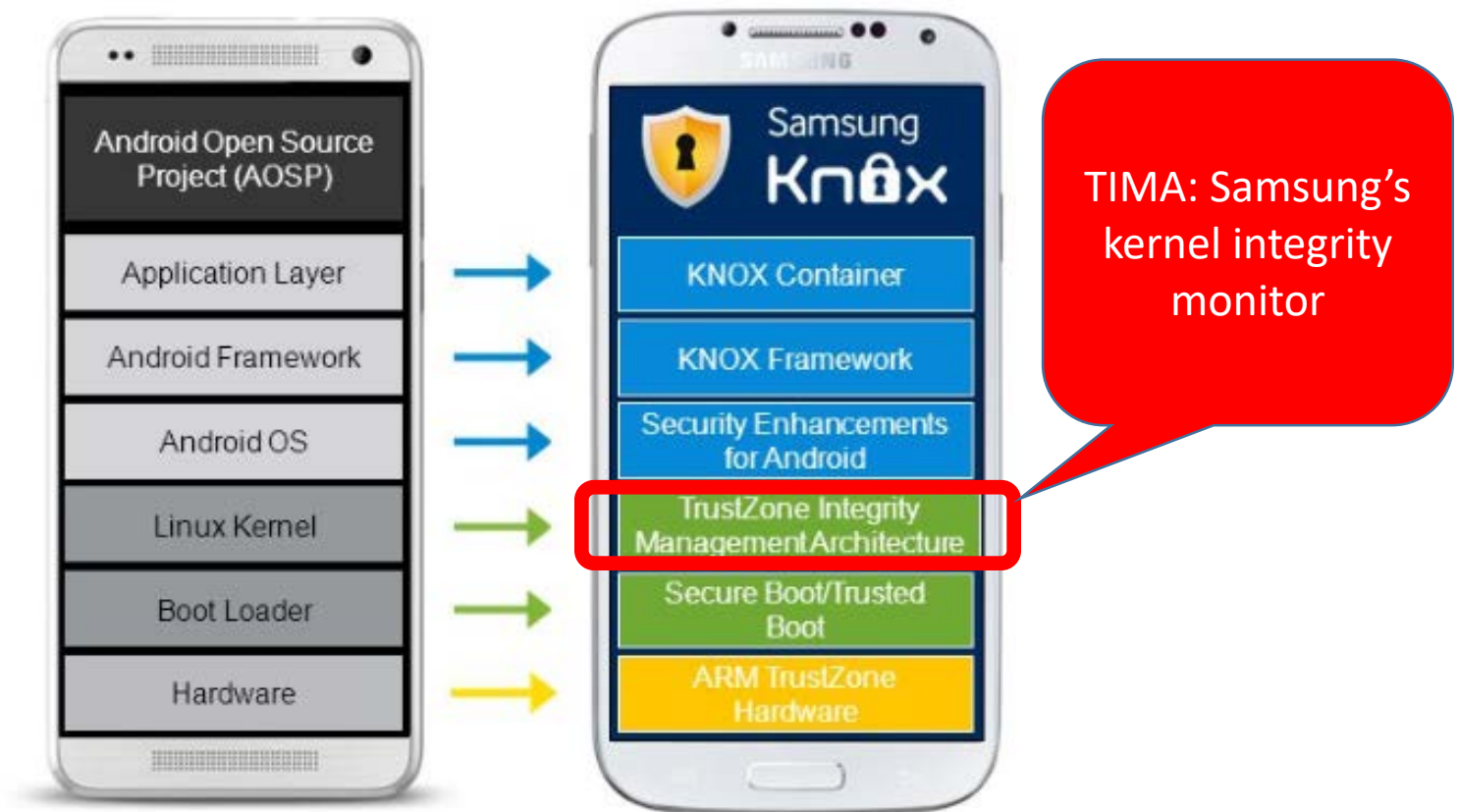

Electric IoT devices

- Define the critical resources and isolate them in the TEE
 - Example: Network components (non-critical), Electric sensor and crypto logic (critical)



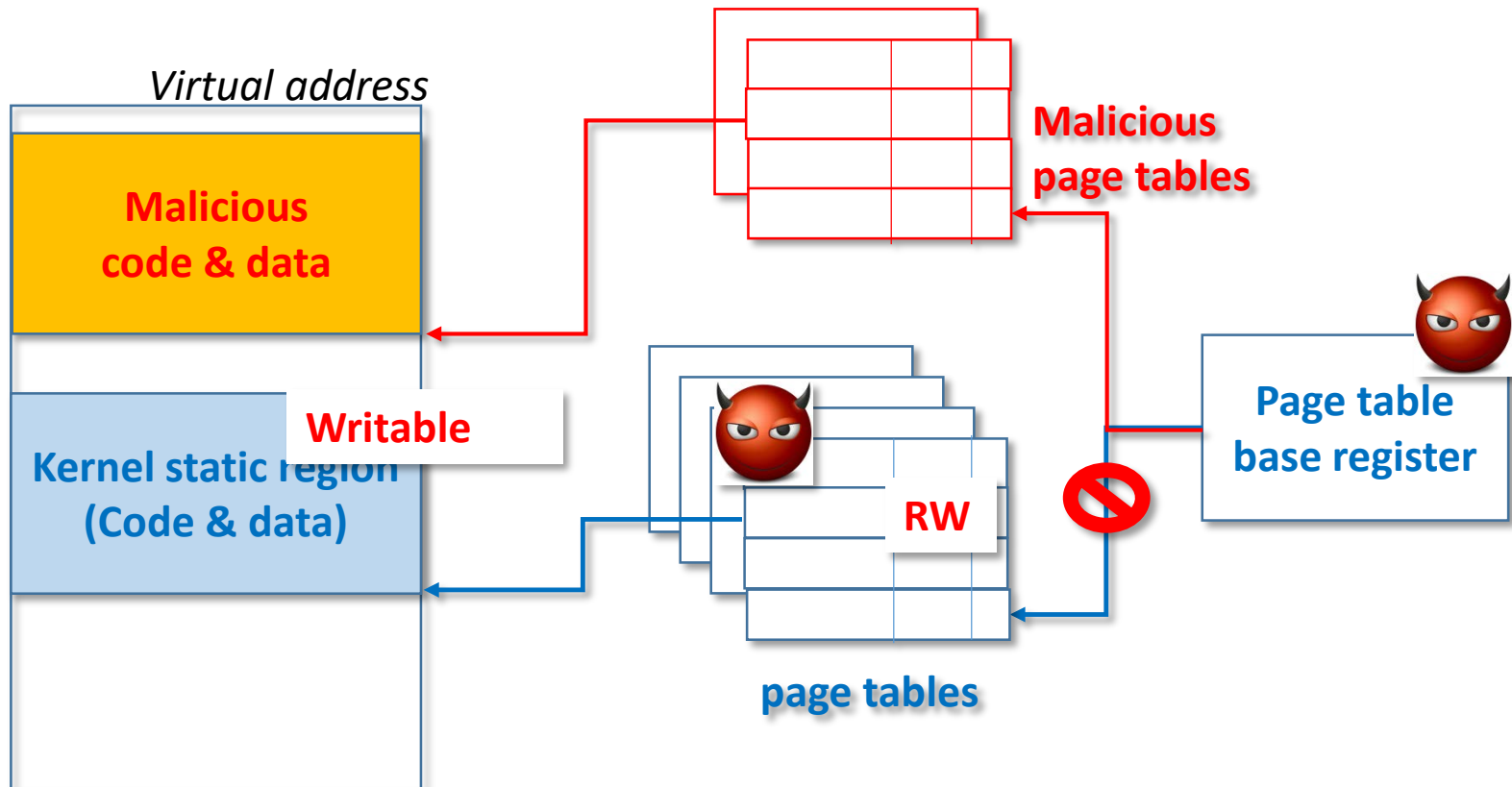
Samsung Mobile Security

❖ Samsung Knox



Case Study: SAMSUNG TIMA

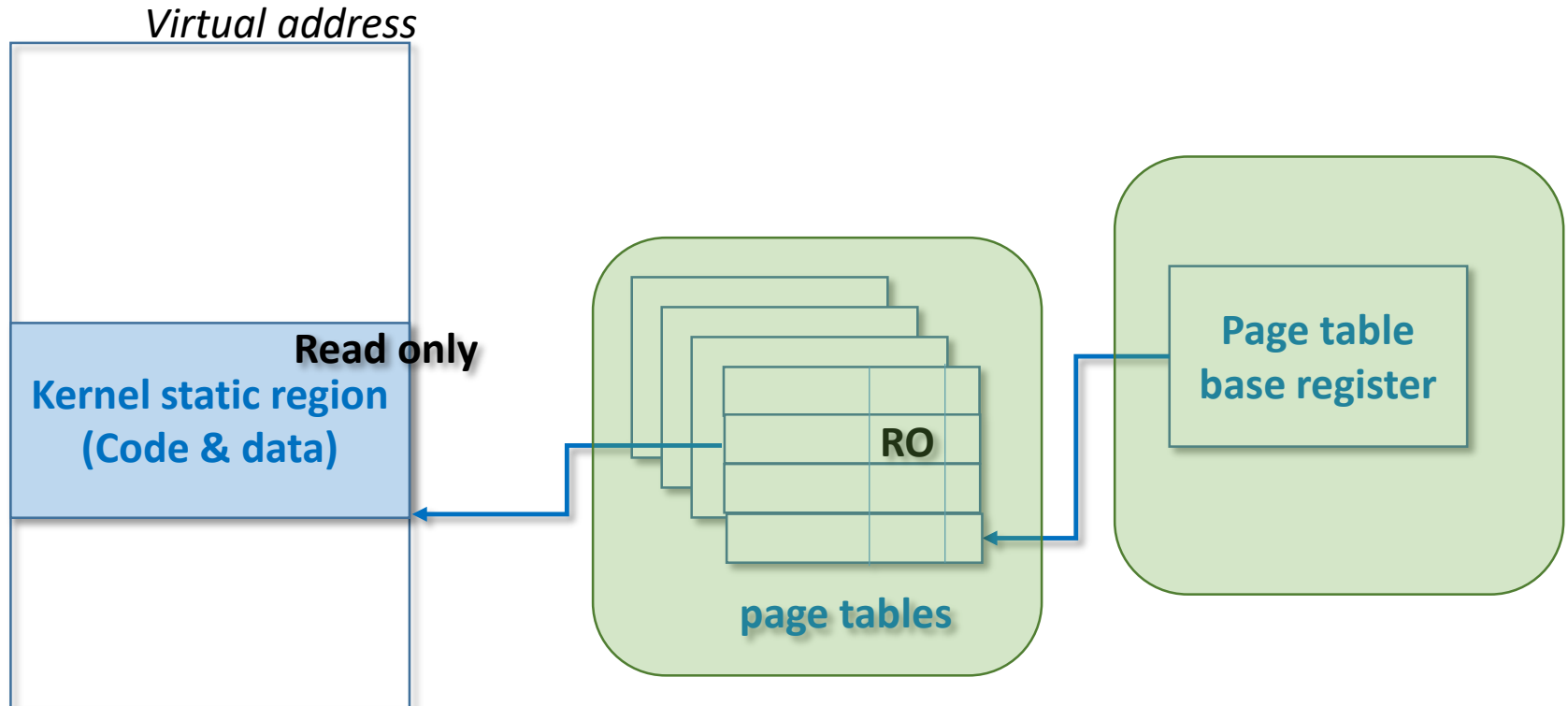
Security critical components.



Case Study: SAMSUNG TIMA

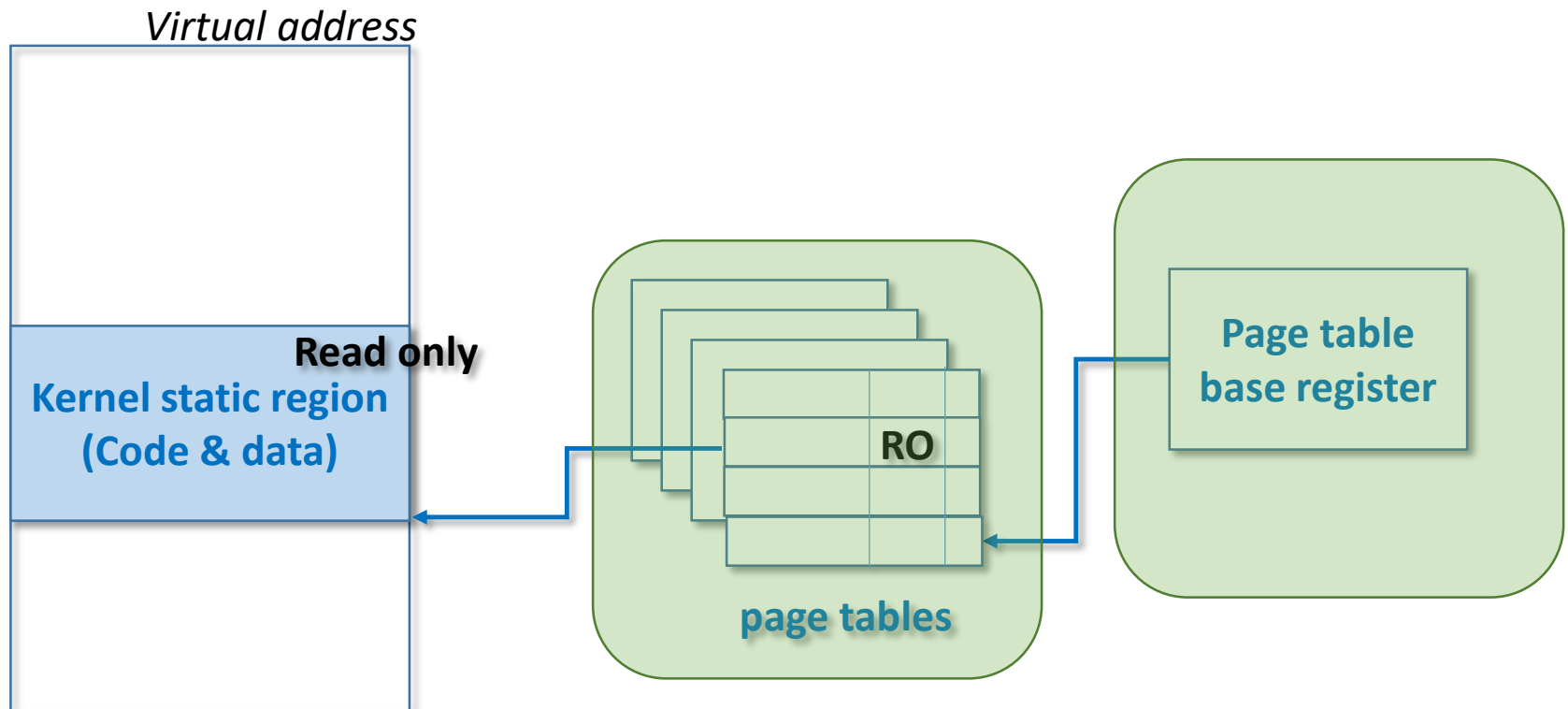
Security critical components.

- Page tables
- System registers (e.g., page table base register)



Case Study: SAMSUNG TIMA

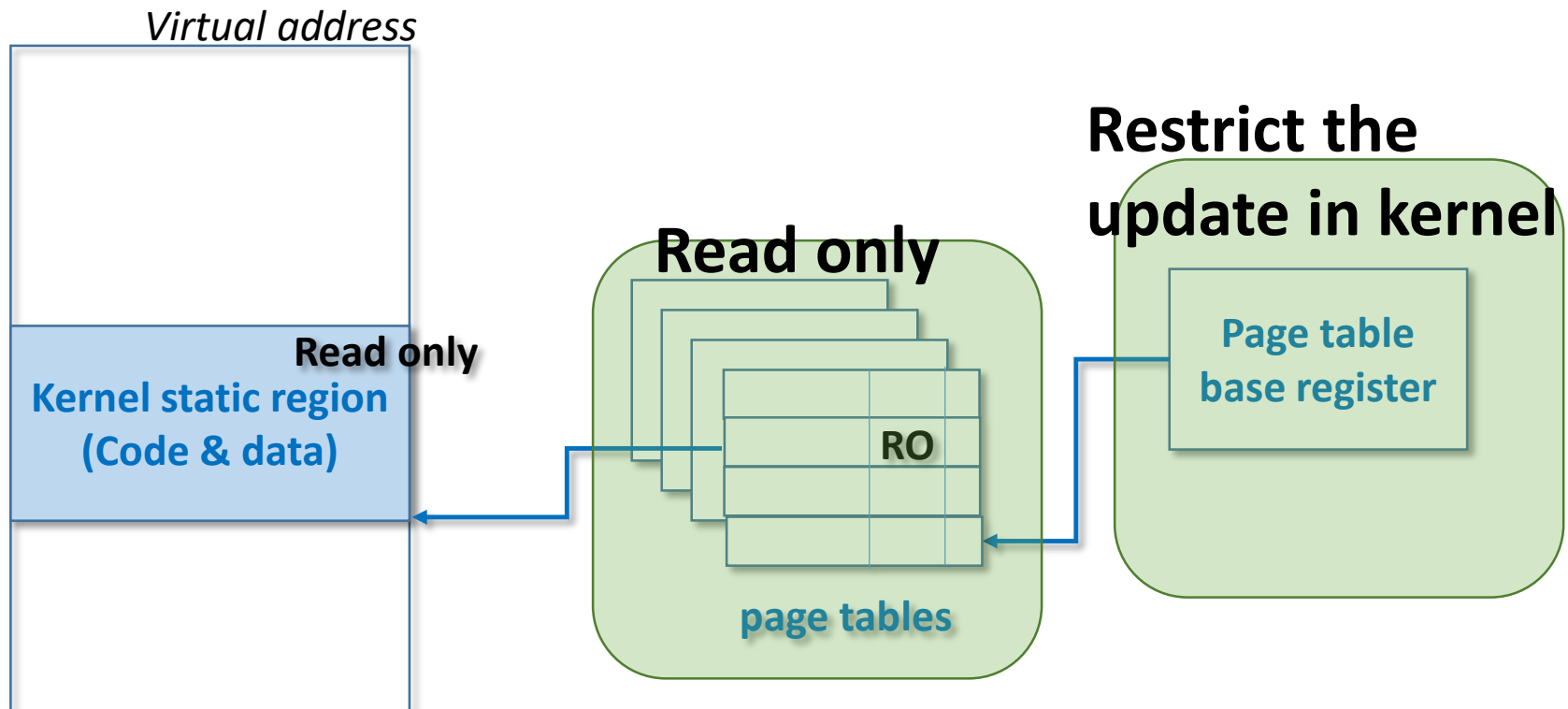
How can we protect them?



Case Study: SAMSUNG TIMA

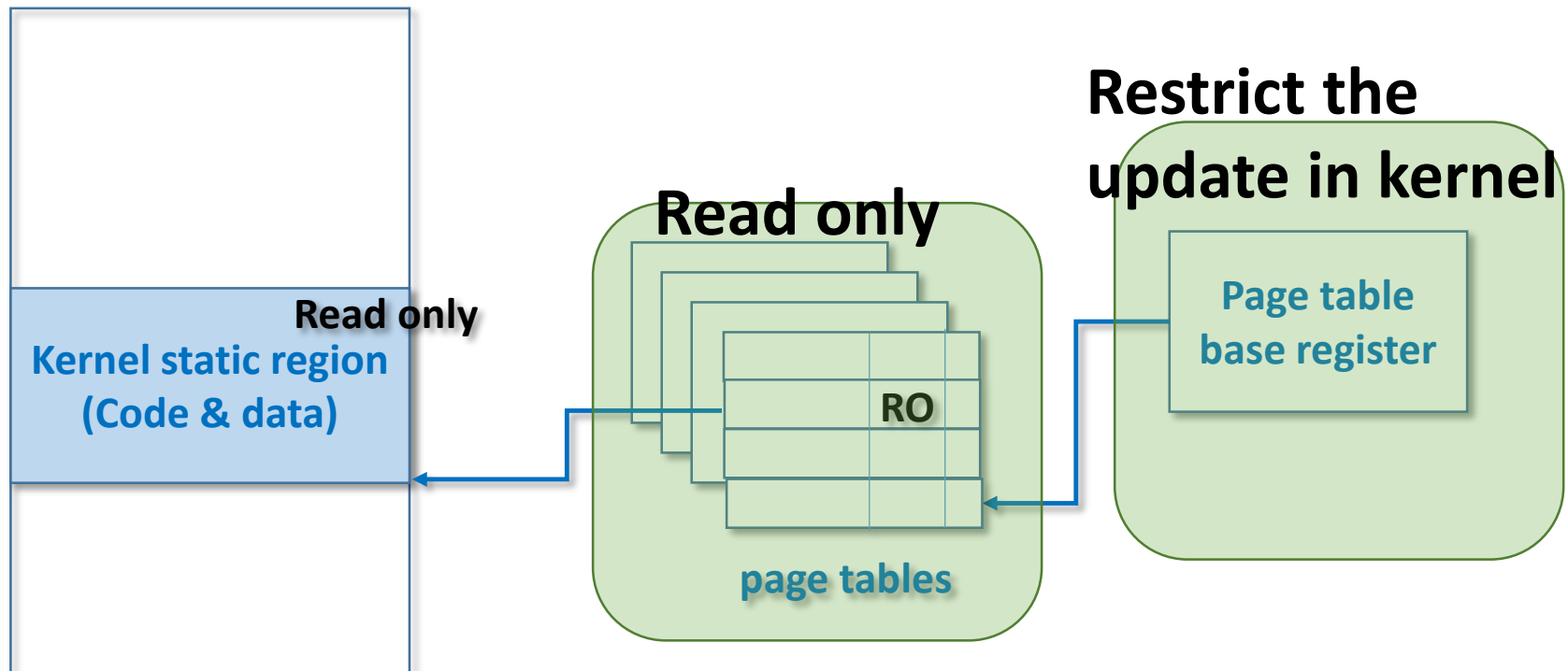
How can we protect them?

→ De-privileging the kernel



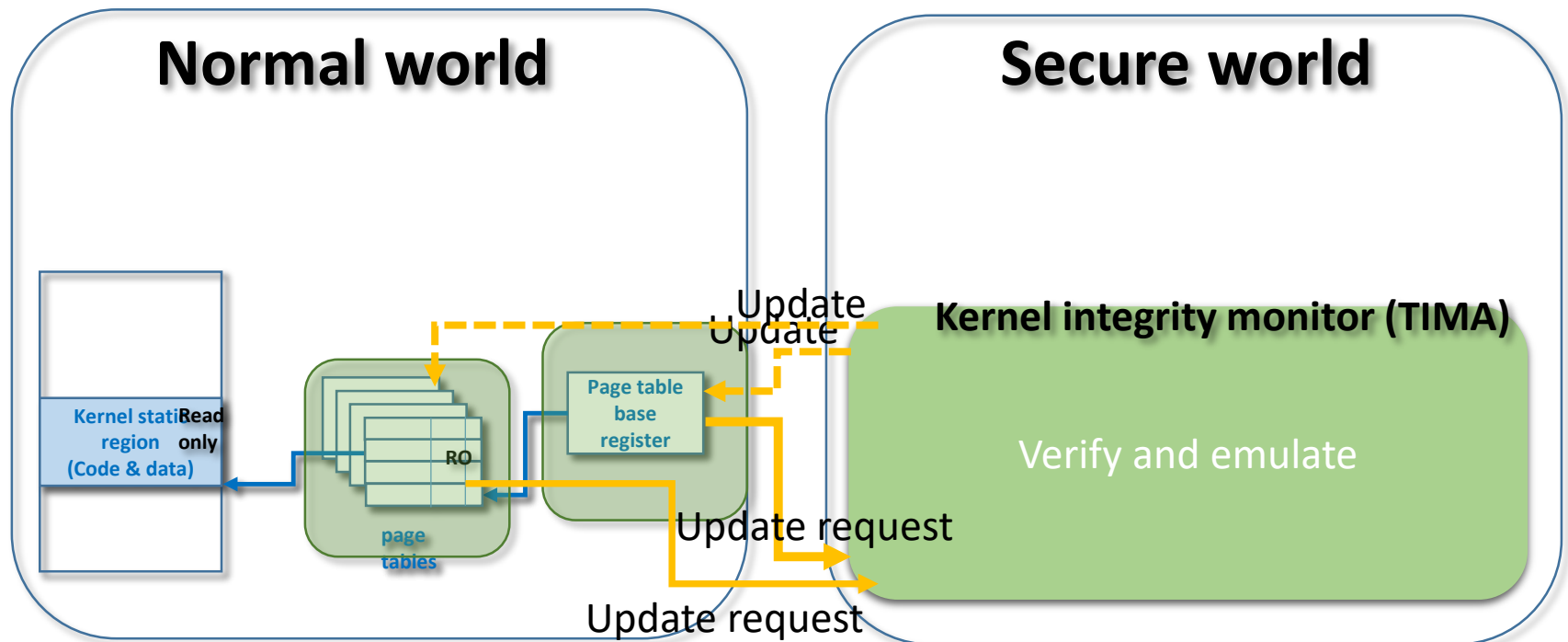
Case Study: SAMSUNG TIMA

How can we support original operations? → Adopt TrustZone



Case Study: SAMSUNG TIMA

TIMA verifies and emulates the security critical operations!



Q & A

Thank you!