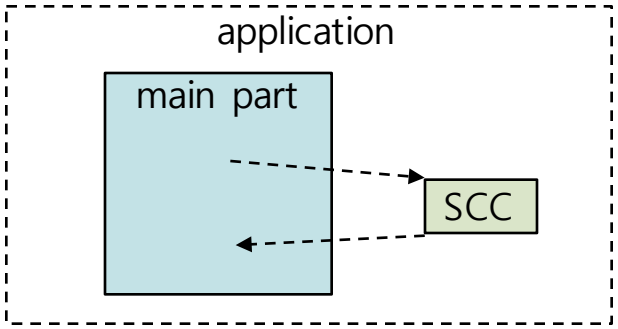
# Hardware-Assisted On-Demand Hypervisor Activation for Efficient Security Critical Code Execution on Mobile Devices

Yeongpil Cho[1] Junbum Shin[2], Donghyun Kwon[1],
MyungJoo Ham2[2], Yuna Kim[2], Yunheung Paek[1]

[1]Seoul National University, [2]Samsung Electronics

# Security Critical Code (SCC)

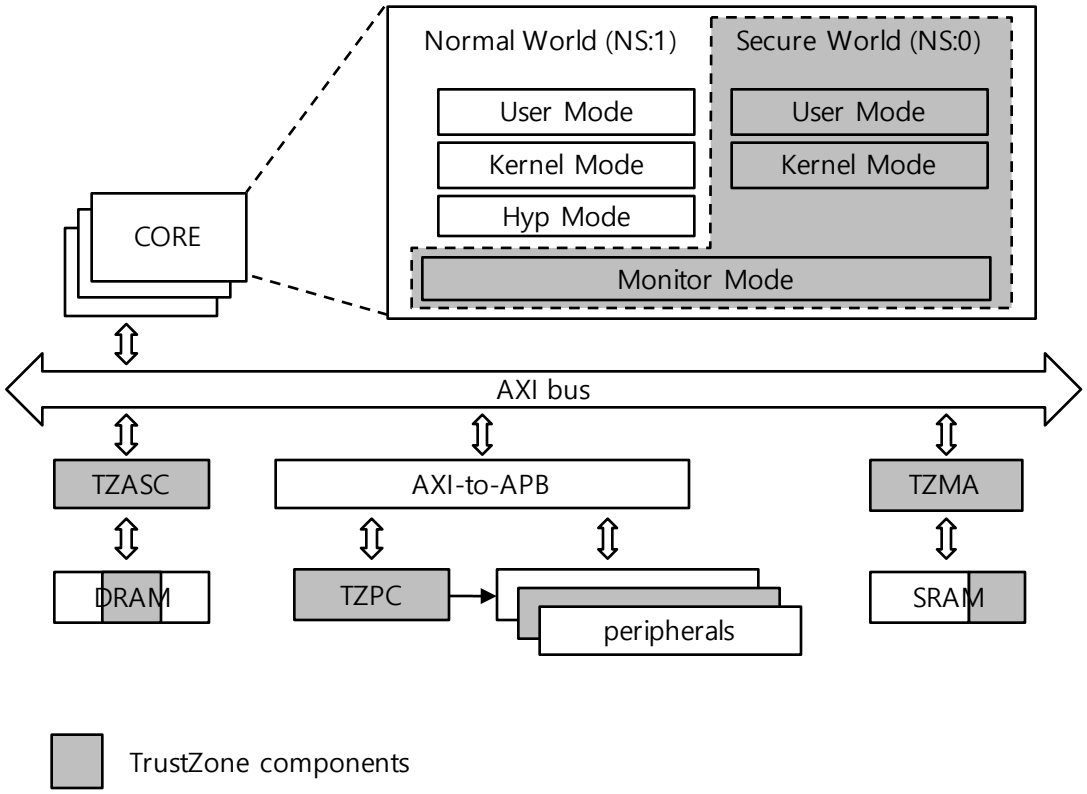
- Applications holds some sensitive data
  - Personal information
  - cryptographic key
- Privilege separation
  - Applications are divided into SCCs and the remainder
  - Only SCCs handle sensitive data




- How can we protect SCCs even if the OS is compromised?
  - We need Trusted Computing Environment (TEE)
- How can we build a TEE on mobile devices?
  - We can rely on ARM TrustZone, as our first choice

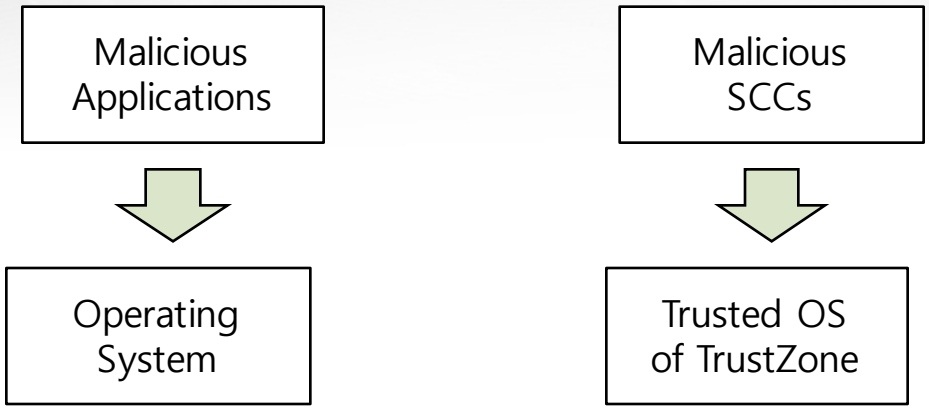
# ARM TrustZone

- Separates system resources into
  - Normal world (less privileged)
  - Secure world (more privileged)
    - secure world processor mode
    - TZASC
      - DRAM
    - TZPC
      - Peripherals
    - TZMA
      - SRAM



# Limitation of TrustZone

- TrustZone is the Trusted Computing Base of the entire System




- Only authorized applications are permitted to run in the existing TEE of TrustZone
  - Vendor's applications
  - OEM applications
- Smartphone Vendors reluctant to open the TEE of TrustZone to 3rd-party developers

# Alternative Approach: μ-Hypervisor

- A number of TEEs based on μ-Hypervisor
  - AppSec, VEE 2015
  - MiniBox, ATC 2014
  - InkTag, ASPLOS 2013
  - TrustVisor, S&P 2010
- Drawback of hypervisor-based approaches
  - a waste of computation power by the complicated address translation
    - relatively high TLB miss penalty

| | Base Native | Nested Paging | Shadow Paging | Agile Paging |
|---|---|---|---|---|
| **TLB hit** | fast (VA⇒PA) | fast (gVA⇒hPA) | fast (gVA⇒hPA) | fast (gVA⇒hPA) |
| **Max. memory access on TLB miss** | 4 | 24 | 4 | ~(4—5) avg. |
| **Page table updates** | fast direct | fast direct | slow mediated by VMM | fast direct |
| **Hardware support** | 1D page walk | 2D+1D page walk | 1D page walk | 2D+1D page walk with switching |

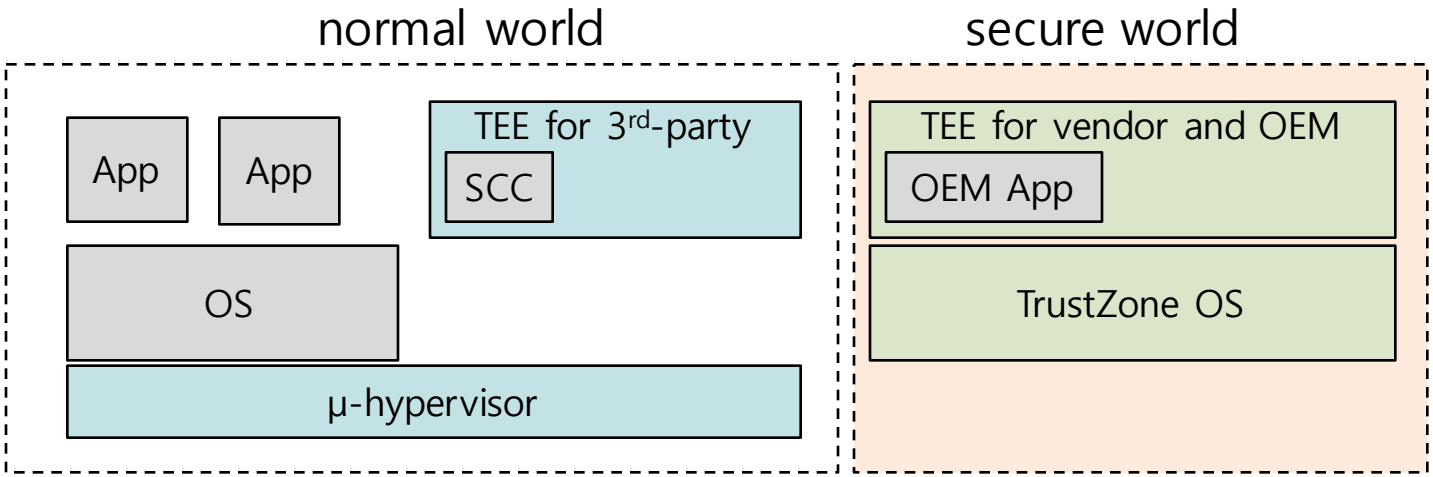
ref: Agile Paging: Exceeding the Best of Nested and Shadow Paging , ISCA 2016

# Our Objectives and Solutions

- We want to provide secure execution environments for 3rd-party developers
  - TrustZone-Hypervisor Hybrid Approach
- We want to minimize the performance impact
  - Dynamic Hypervisor Activation Scheme

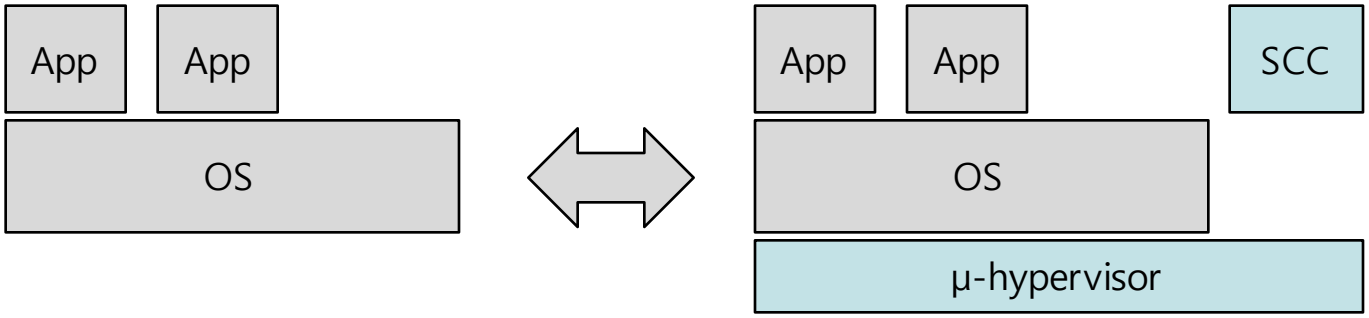
# Our Solution 1: Hybrid Approach

- Bimodal TEEs
  - The existing TEE of TrustZone
    - for vendor and OEM applications
  - Alternative TEE based on μ-hypervisor
    - for 3rd-party developers and their SCCs
- Alternative TEE might be compromised by malicious SCCs
  - But, we can prevent the damage from spreading to the secure world



# Our Solution 2: Dynamic Activation Scheme

- We use μ-hypervisor to build a TEE
- To minimize the performance overhead, we activate the μ-hypervisor only when it is needed
  - an SCC account for a small fraction of the entire application

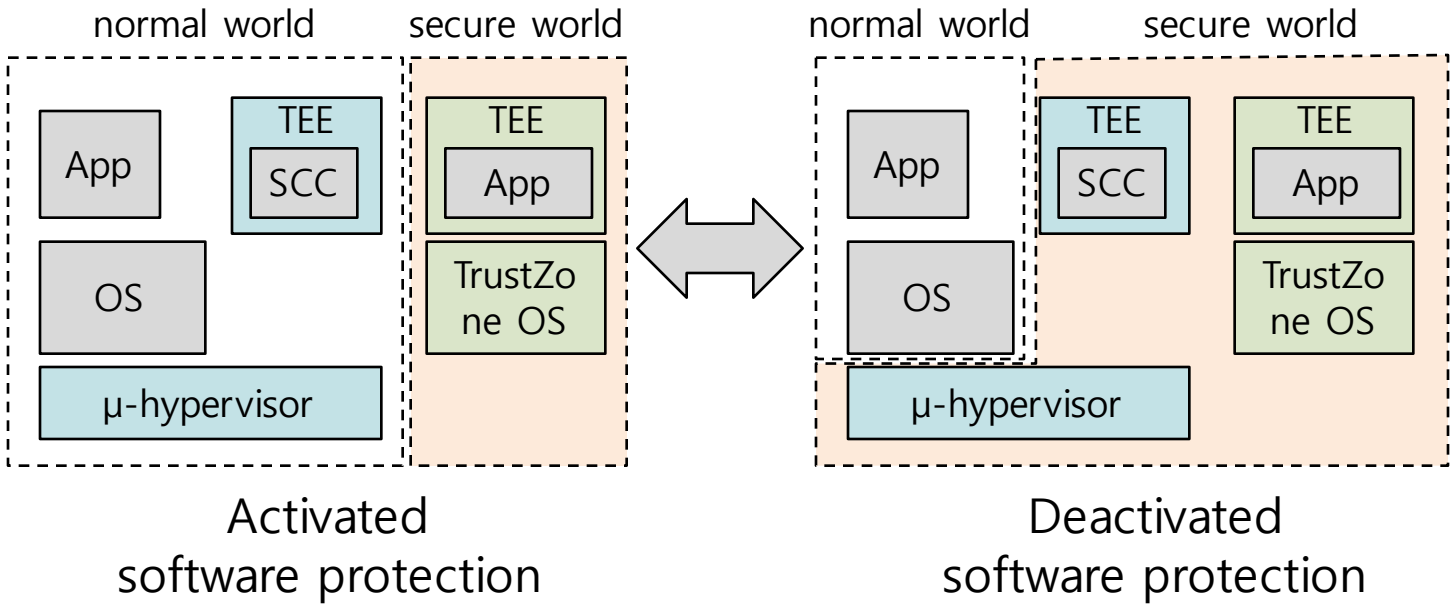



Deactivated software protection

Activated software protection

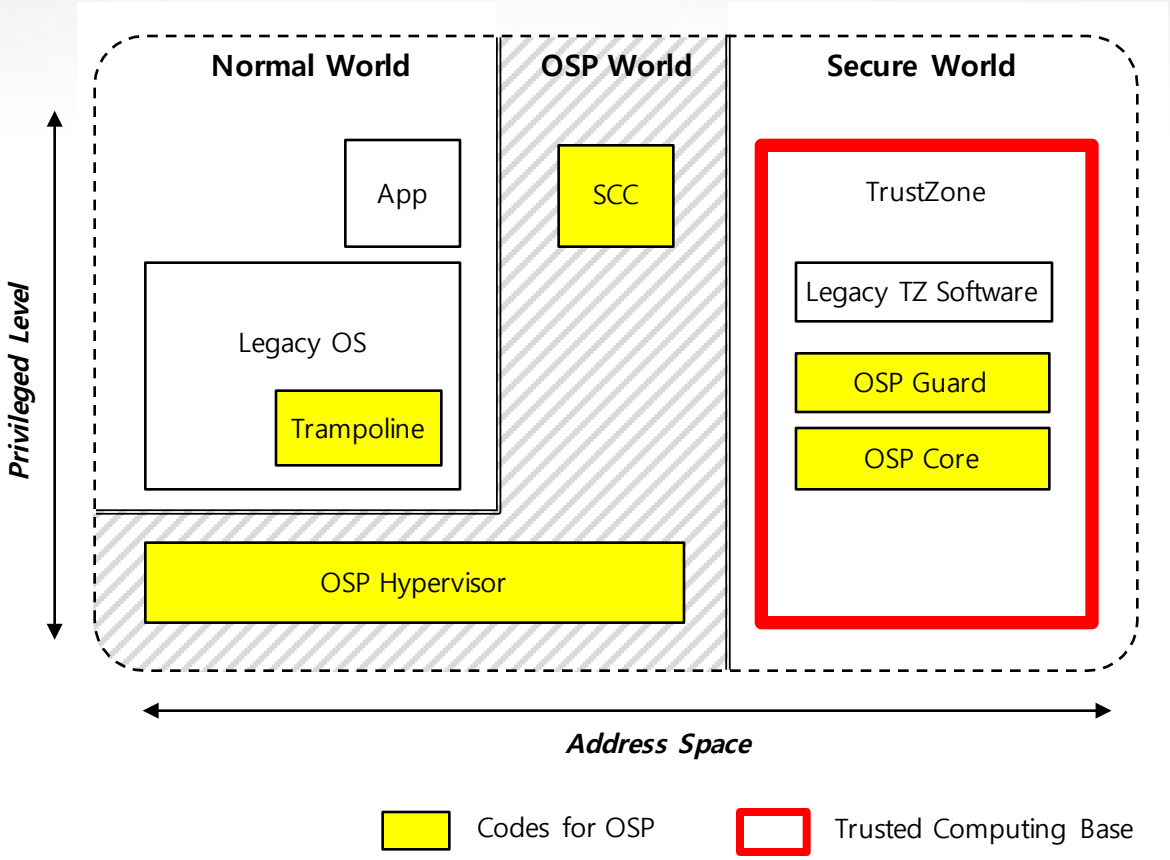
# Our Solution 2: Dynamic Activation Scheme

- Our strategy for protecting sensitive states of μ-hypervisor and SCCs
  - while the μ-hypervisor is activated
    - by using the μ-hypervisor
  - while the μ-hypervisor is deactivated
    - by covering with the secure world



# On-demand Software Protection (OSP)

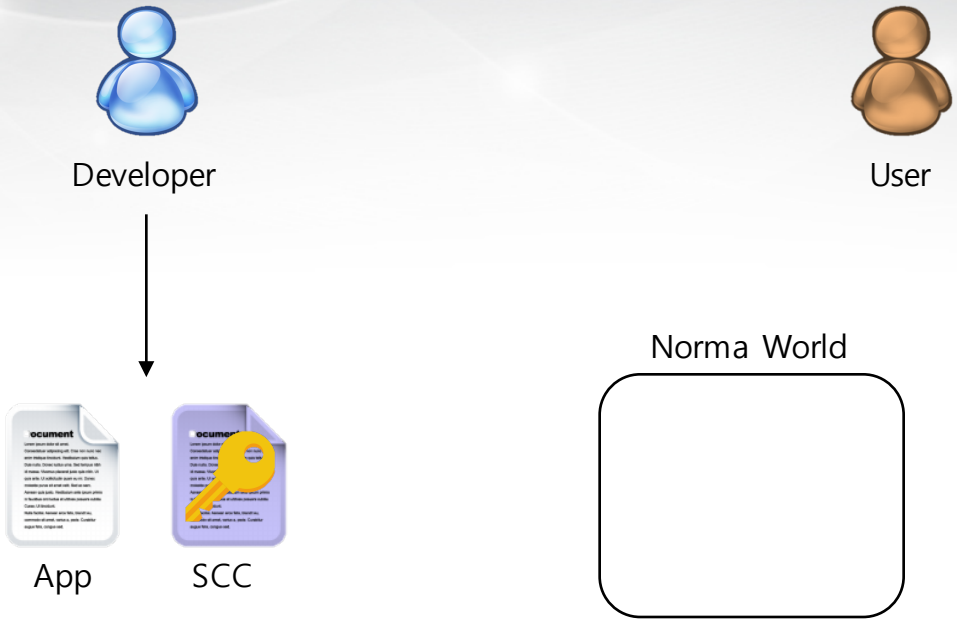
## Overall architecture



# Components

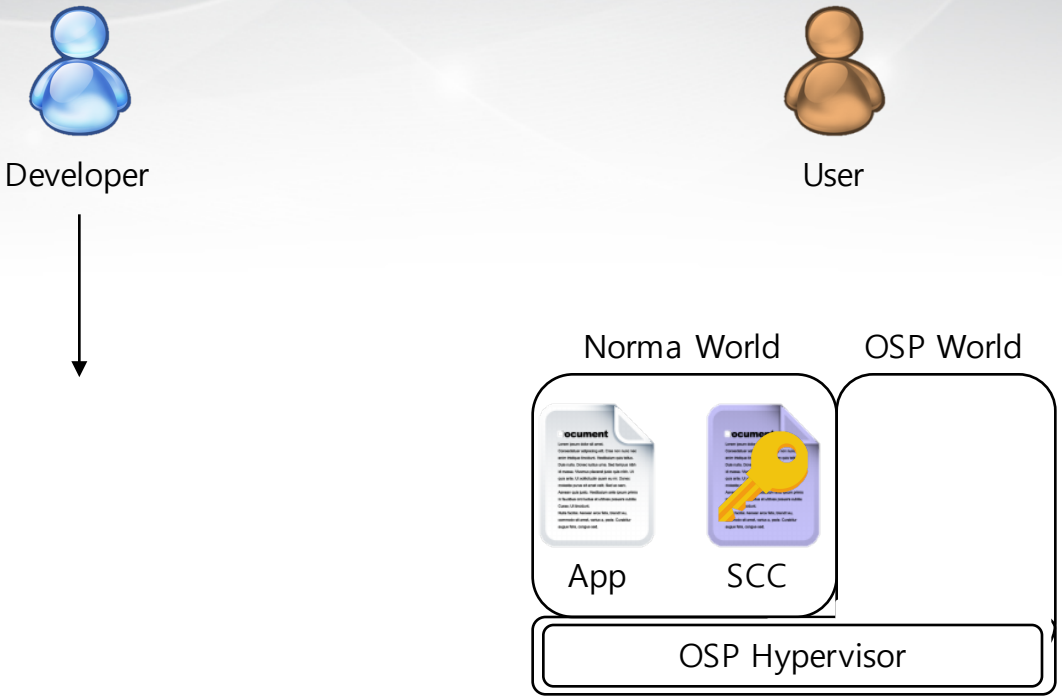
- Secure world components
  - OSP Core
    - initializes and controls OSP hypervisor and TrustZone components
  - OSP Guard
    - a set of cryptographic functions and key management functions
- OSP world component
  - OSP Hypervisor
    - provides secure execution environments for SCCs
- Normal world component
  - Trampoline
    - provides applications with the interface that can communicate with OSP

# Protection of an SCC



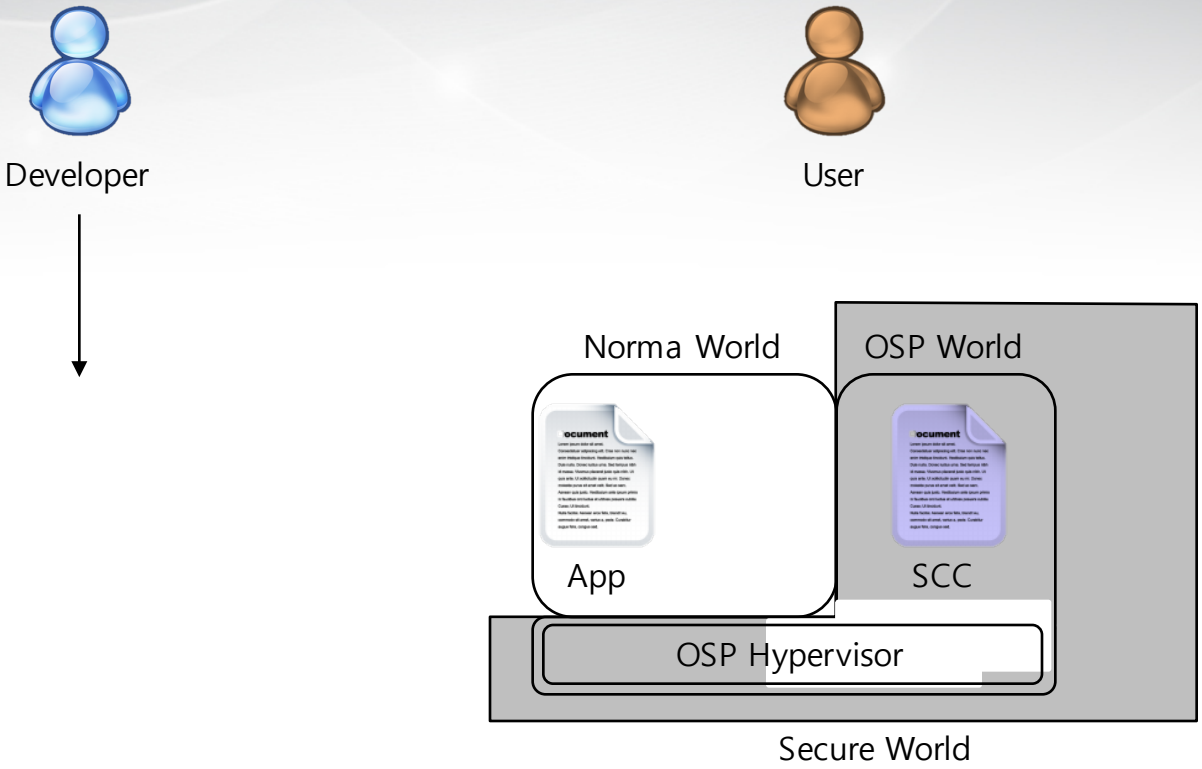

- Application Development
  - A developer develops and uses an SCC like an external library
  - An SCC will be distributed after being encrypted with the PK_osp

# Protection of an SCC




- Execution of an SCC
  - Under the protection of OSP hypervisor, an SCC is decrypted and executed according to the lifecycle model of SCC

# Protection of an SCC




- Deactivation of OSP hypervisor
  - if there is no SCC running in the OSP hypervisor, the hypervisor is deactivated and is protected by being included into the secure world

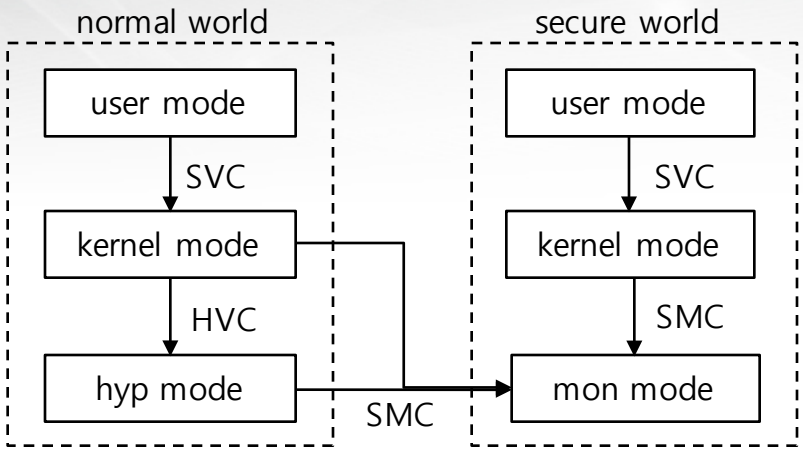
# Programming Interfaces of an SCC

- OSP follows the SCC lifecycle model of TrustVisor
- Management and Service Interfaces

| Function Name | Parameter | Call-site | Description |
|---|---|---|---|
| **Management interfaces** | | | |
| SCC_register | scc_file_name, ptr_external_handler | app | Registers an SCC with a specification. Upon success, returns the SCC's number. |
| SCC_unregister | scc_num | app | Unregisters an SCC. |
| SCC_parameter_add | ptr_scc_param_spec, param_flag, ptr_param, length | app | Add a parameter to a parameter specification. |
| SCC_invoke | scc_num, entry_func, ptr_scc_param_spec, arg0...arg3 | app | Invokes an SCC with a parameter specification. Upon finish, returns a return value. |
| SCC_ret_to_scc | scc_num, return_value | app | Return to an SCC with a return value |
| **Service interfaces** | | | |
| OSP_save | ptr_data, length | SCC | Save data on secure storage. Upon success, returns the storage number. |
| OSP_load | storage_num, ptr_buffer, length | SCC | Loads the data for a storage number. |
| OSP_delete | storage_num | SCC | Deletes the data for a storage number. |
| OSP_encrypt | ptr_data, ptr_buffer, length | SCC | Encrypt data |
| OSP_decrypt | ptr_data, ptr_buffer, length | SCC | Decrypt data |
| OSP_signing | ptr_data, length, private_key, signature | SCC | Sign data with a given private key |
| OSP_verification | ptr_data, length, public_key, signature | SCC | Verify data with a given public key |
| OSP_external_handler | cmd, arg0...arg3 | SCC | Call the external handler with parameters. Upon finish, returns a return value |

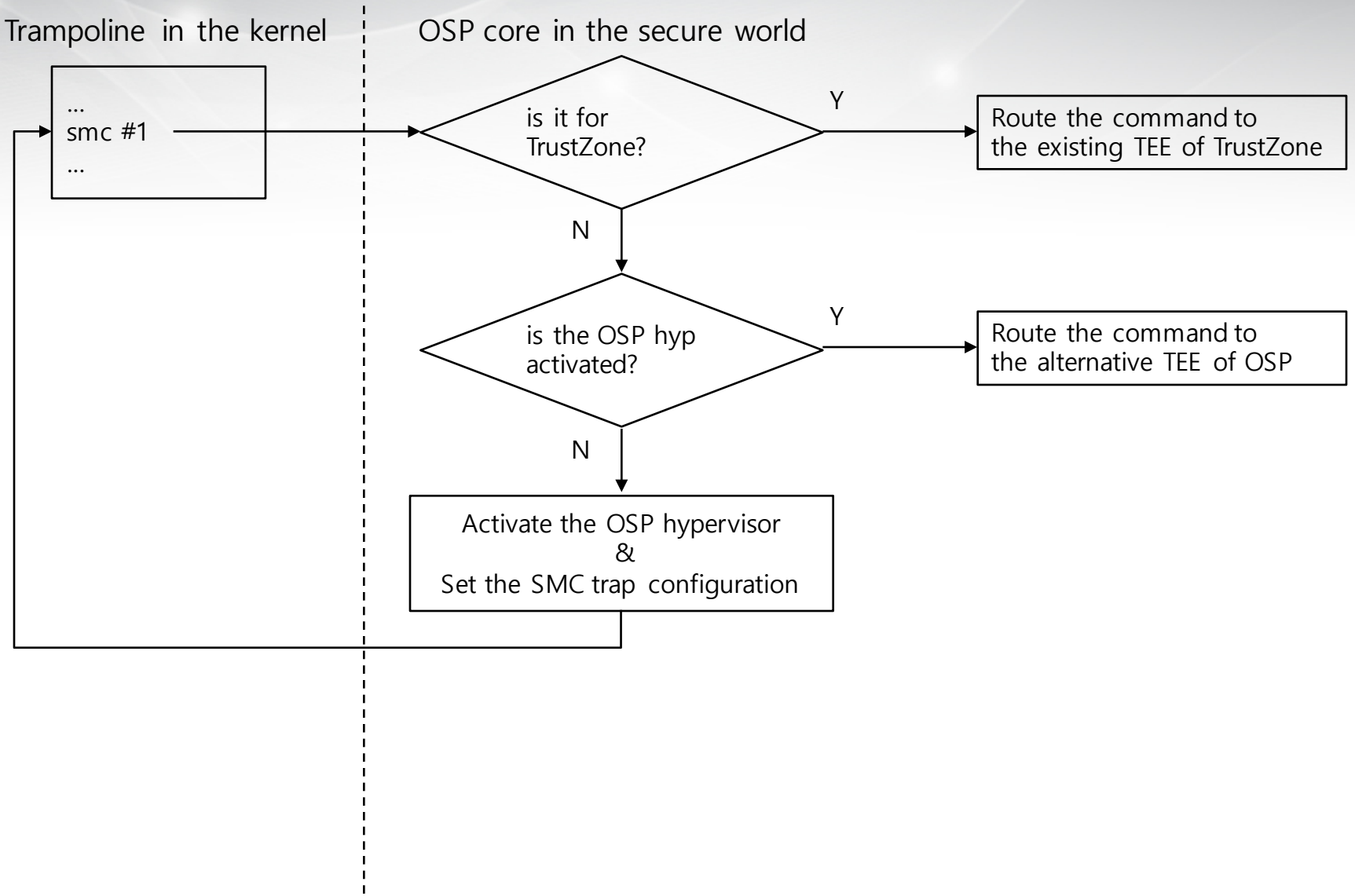
# Interface to an SCC

- ARM instructions to cross over the privilege boundaries

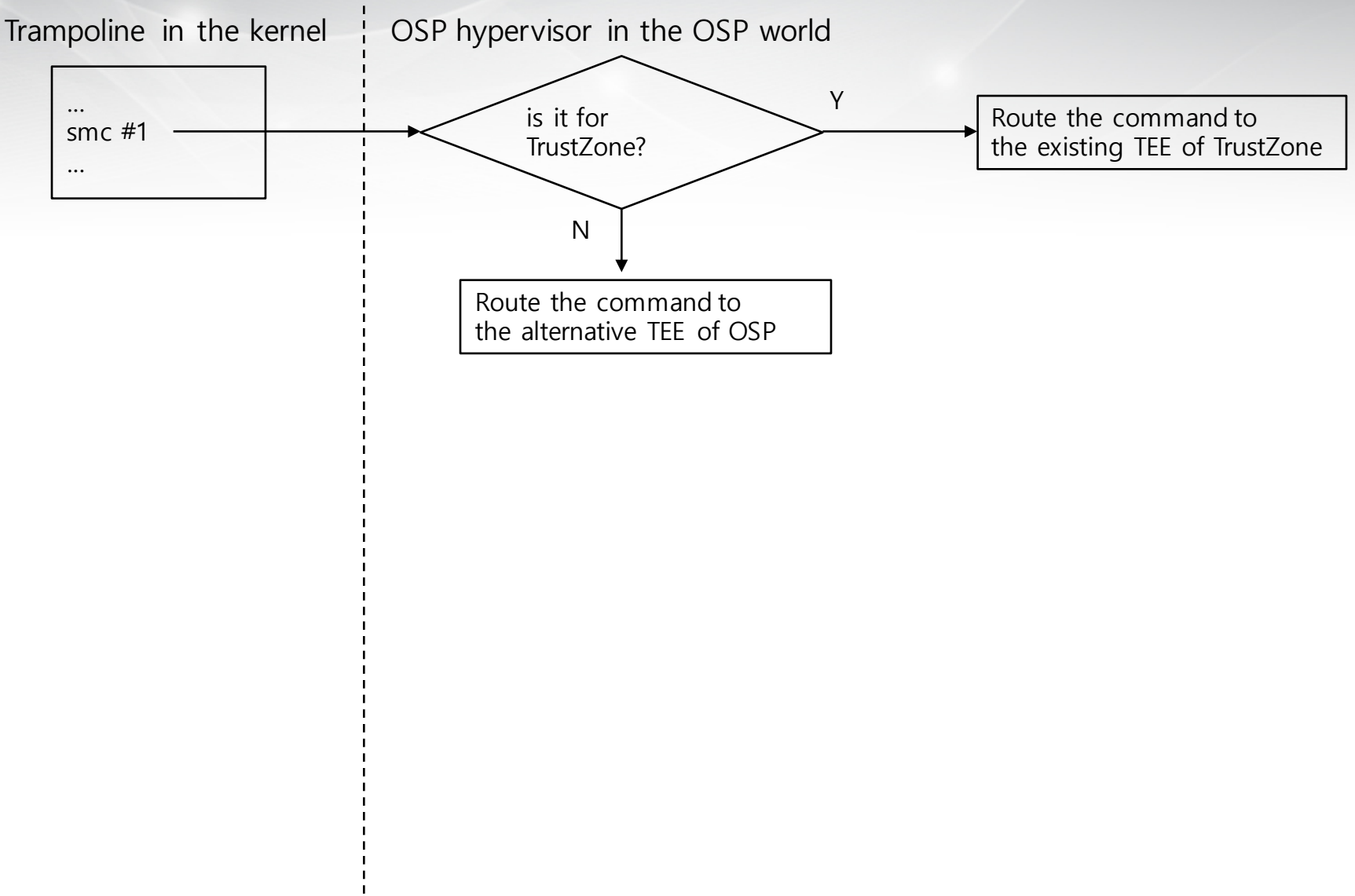



- For interfacing, Trampoline, in the kernel, needs to execute
  - when the OSP hypervisor is activated ➔ HVC instruction
  - when the OSP hypervisor is deactivated ➔ SMC instruction
    - OSP core in the secure world will activate the OSP hypervisor
- But OSP only uses the SMC to construct the unified interface
  - when the OSP hypervisor is activated ➔ SMC instruction
    - set TSC-bit of HCR to make the SMC instruction be trapped into the hyp mode
  - when the OSP hypervisor is deactivated ➔ SMC instruction

# Interface to an SCC

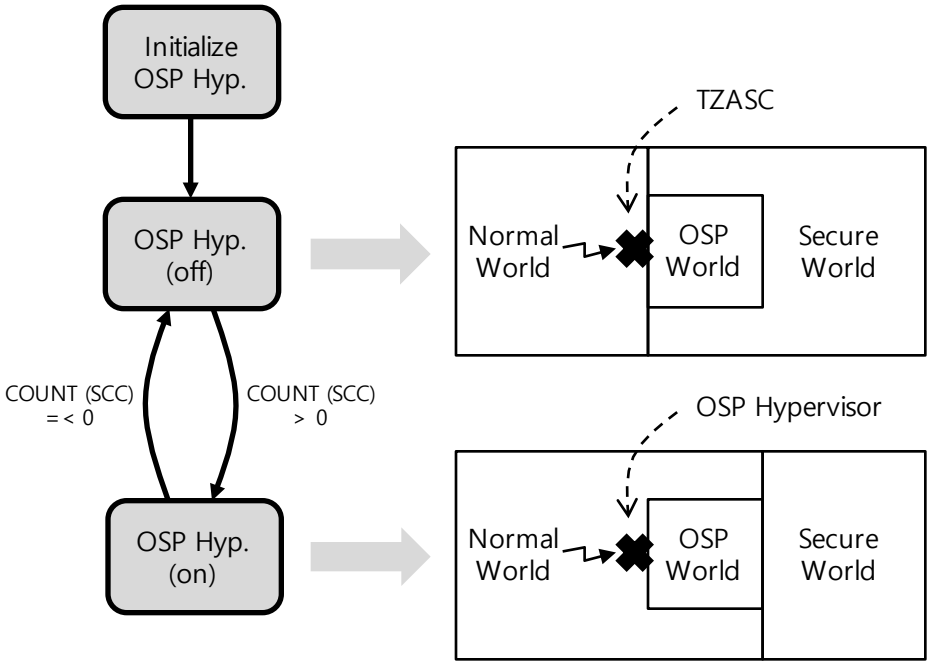


# Interface to an SCC



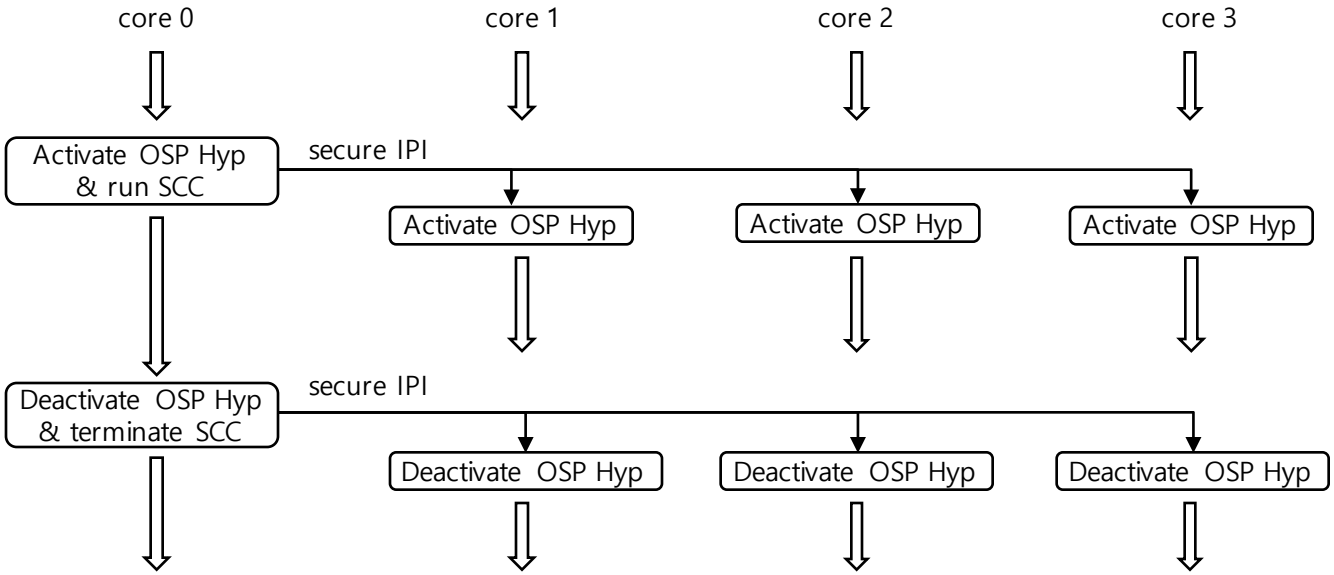
# Protection of OSP World

- OSP uses bimodal protection schemes based on two hardware features
  - TZASC of TrustZone
  - extended paging of the OSP hypervisor
- It depends on the number of SCCs running in the OSP world concurrently



# Multi-core Support

- In a multi-core environment
  - Each core has its own MMU supporting extended paging
  - Every core shares one TZASC, which is located in between AXI bus and DRAM
- Synchronization Problem
  - ex) when a core activates the OSP hyp, if another doesn't activate that…



# Dynamic activation routine

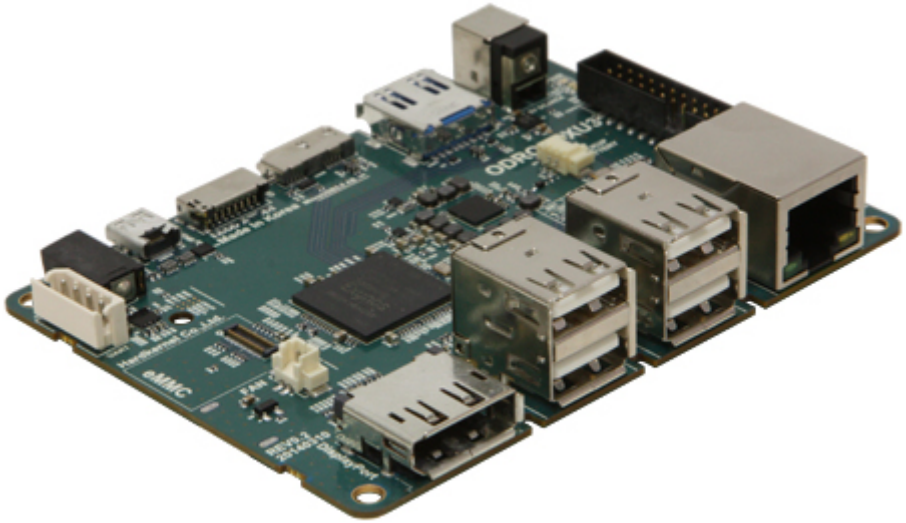
- OSP core, in the secure world, can
  - control the extended paging by using privileged instructions
  - control TZASC by using memory-mapped registers
- To prevent sensitive data of the OSP world from being disclosed by cache-poisoning attack
  - cache entries corresponding to the memory region of the OSP world should be cleaned and invalidated

**Procedure ACTIVATE_OSP_HYP**
Enable the extended paging
Send secure IPIs to other cores to enable the extended paging, too
Reduce the secure world to reveal the OSP world using TZASC
**End**

**Procedure DEACTIVATE_OSP_HYP**
Expand the secure world to cover the OSP world using TZASC
Clean and invalidate cache entries of the OSP world
Disable the extended paging
Send secure IPIs to other cores to disable the extended paging, too
**End**

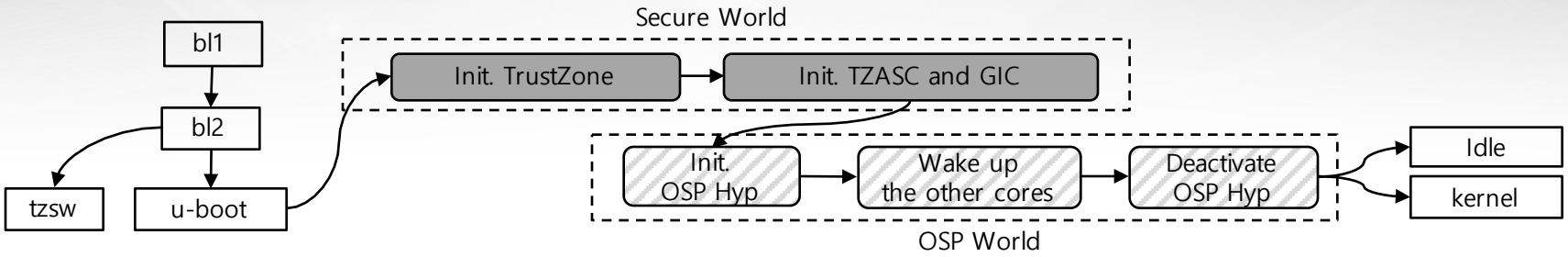
# Implementation

- ODROID-XU3-LITE
  - exynos 5422
    - Cortex A15 1.8GHz quad core
    - Cortex A7 1.3GHz quad core
  - 2 GB RAM
- OS
  - Android 4.4.2 with Linux Kernel 3.10



# Boot-up sequence

- The OSP core gets control before the kernel starting




- The OSP core must be executed in the secure world
  - replaces the last instruction of the u-boot to a specific 'smc' instruction
  - modifies the smc handler of tzsw to transfer control to the OSP core in the secure world
- Top 128 MBytes are allocated for OSP


- Disables bigLITTLE feature
  - OSP only utilizes four big cores

# OSP Boot-logs

u-boot

the OSP core

kernel

```
>>> Load Boot Script from mmc 0:1 <<<
reading boot.scr

** Unable to read "boot.scr" from mmc 0:1 **
>>> Load Boot Script from mmc 0:2 <<<

** Unable to use mmc 0:2 for fatload **
>>> Run Default Bootcmd <<<
reading kernel..device 0 Start 1263, Count 16384
MMC read: dev # 0, block # 1263, count 16384 ... 16384 blocks read: OK
completed
reading RFS..device 0 Start 17647, Count 2048
MMC read: dev # 0, block # 17647, count 2048 ... 2048 blocks read: OK
completed

Starting kernel ...

[c0] start to boot
[c0] page_initialize_lv1_table: setting up 1st level hyp translation tables...
[c0] init OSP hypervisor
[c0] init OSP hypervisor in non-secure Monitor mode
[c0] page_initialize_lv1_table: setting up 1st level hyp translation tables...
[c0] page_initialize_lpae_table: setting up 2nd level translation tables...
[c0] page_initialize_lpae_table: 2nd level translation tables initialized.
[c0] turn on core 1
[c1] start to boot
[c1] page_initialize_lv1_table: setting up 1st level hyp translation tables...
[c1] init OSP hypervisor
[c1] init OSP hypervisor in non-secure Monitor mode
[c0] turn on core 2
[c2] start to boot
[c2] page_initialize_lv1_table: setting up 1st level hyp translation tables...
[c2] init OSP hypervisor
[c2] init OSP hypervisor in non-secure Monitor mode
[c0] turn on core 3
[c3] start to boot
[c3] page_initialize_lv1_table: setting up 1st level hyp translation tables...
[c3] init OSP hypervisor
[c3] init OSP hypervisor in non-secure Monitor mode
[c0] start to the kernel
[c0] deactivate OSP hypervisor
[c2] deactivate OSP hypervisor
[c3] deactivate OSP hypervisor
[c1] deactivate OSP hypervisor
[    0.000000] [c0] Booting Linux on physical CPU 0x0
[    0.000000] [c0] Initializing cgroup subsys cpu
[    0.000000] [c0] Initializing cgroup subsys cpuacct
[    0.000000] [c0] Linux version 3.10.9 (dhkwon@sorcloud1-System-Product-Name) (gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5) ) #115 S
[    0.000000] [c0] CPU: ARMv7 Processor [412fc0f3] revision 3 (ARMv7), cr=10c5387d
```

# SCC running-log

SCC registration

SCC invocation

SCC unregistration

```
[c0] activate OSP hypervisor
[c3] activate OSP hypervisor
[c2] activate OSP hypervisor
[c1] activate OSP hypervisor
[c0] hyp-call is raised
[c0] (SCC:0) regist
[c0] deactivate OSP hypervisor
[c2] deactivate OSP hypervisor
[c1] deactivate OSP hypervisor
[c3] deactivate OSP hypervisor
[c1] activate OSP hypervisor
[c0] activate OSP hypervisor
[c2] activate OSP hypervisor
[c3] activate OSP hypervisor
[c1] hyp-call is raised
[c1] (SCC:0) invoke
[c1] (SCC:0) parameter marshalling
[c1] (SCC:0) scc start
[c1] hyp-call is raised
[c1] (SCC:0) svc call
[c1] data:7959b158 len:123 priv_key_con:bbed4fc8
[c1] prefetch abort is raised
[c1] (SCC:0) scc end
[c1] (SCC:0) parameter unmarshalling
[c1] deactivate OSP hypervisor
[c3] deactivate OSP hypervisor
[c0] deactivate OSP hypervisor
[c2] deactivate OSP hypervisor
[c0] activate OSP hypervisor
[c3] activate OSP hypervisor
[c2] activate OSP hypervisor
[c1] activate OSP hypervisor
[c0] hyp-call is raised
[c0] (SCC:0) unregist
[c0] deactivate OSP hypervisor
[c3] deactivate OSP hypervisor
[c2] deactivate OSP hypervisor
[c1] deactivate OSP hypervisor
```

# Evaluation

- Round-trip CPU cycles of dynamic activation and deactivation of the OSP hypervisor

| List | Cycles | Time (at 1.8 GHz) |
|---|---|---|
| Overall | 127,453 | 70.81 us |
| Control transfer between the OSP core and the kernel | 1,990 | 1.11 us |
| Synchronization of the activation state of the OSP hypervisor in the multi-core environment | 11,191 | 6.22 us |
| Cache clean & invalidation | 31,450 | 17.47 us |
| Verification of page tables of the System MMU | 68,329 | 37.96 us |

# Evaluation

- Run-time overhead according to the activation state of the OSP hypervisor

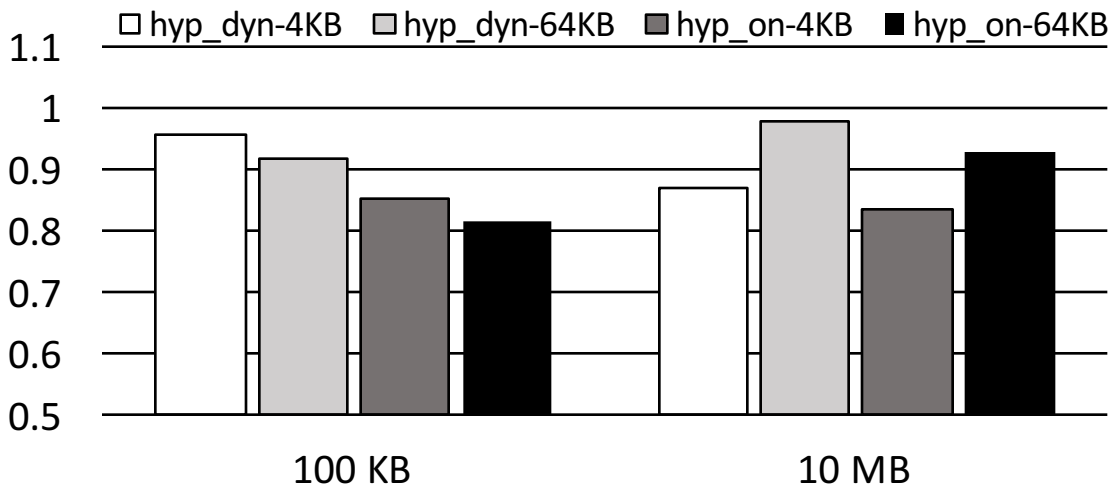
| Benchmark | Performance Overhead | | Energy Overhead | |
|---|---|---|---|---|
| | Deactivation | Activation | Deactivation | Activation |
| CF-bench | 1 % | 3 % | 1 % | 4 % |
| AnTuTu | 0 % | 2 % | 0 % | 3 % |
| Vellamo-Browser | 1 % | 11 % | 0 % | 4 % |
| Vellamo-Machine | -1 % | 5 % | 0 % | 4 % |
| BaseMark | 1 % | 4 % | -1 % | 4 % |
| Geekbench | 0 % | 2 % | 0 % | 2 % |
| iozone-write | 0 % | 3 % | 0 % | 8 % |
| GFXbench-Frames | 1 % | 13 % | -1 % | 4 % |

# Evaluation

- (ID/PW) Autocomplete function of Chromium browser
  - encrypting ID and password with a SCC
  - baseline: Running same code without OSP
- Results
  - loading time for m.facebook.com
    - average: 995.7 ms
    - stddev: 71.6 ms
  - execution time of the SCC doesn't affect the loading time at all
    - average: 0.101 ms
    - very smaller than the stddev of the loading time

# Evaluation

- File encryption applications
  - Performs file encryption in the SCC
  - baseline: Running same code without a hypervisor
- Results
  - A number of SCC invocations (causing dynamic activation) increases the performance overhead
  - Dynamic activation of a hypervisor may improve performance when the hypervisor incurs a considerable performance penalty



# Summary

- How can we provide public developers' SCCs with secure execution environments?
  - ARM TrustZone?
    - Technically possible
    - but, malicious SCC may compromise TrustZone, the TCB of the entire system
- OSP
  - Hypervisor-based alternative TEE for public application developers
  - Performance optimization
    - On-demand activation of the hypervisor
    - Protection of the intermediate states of the hypervisor and SCCs using TrustZone

# Thank you!