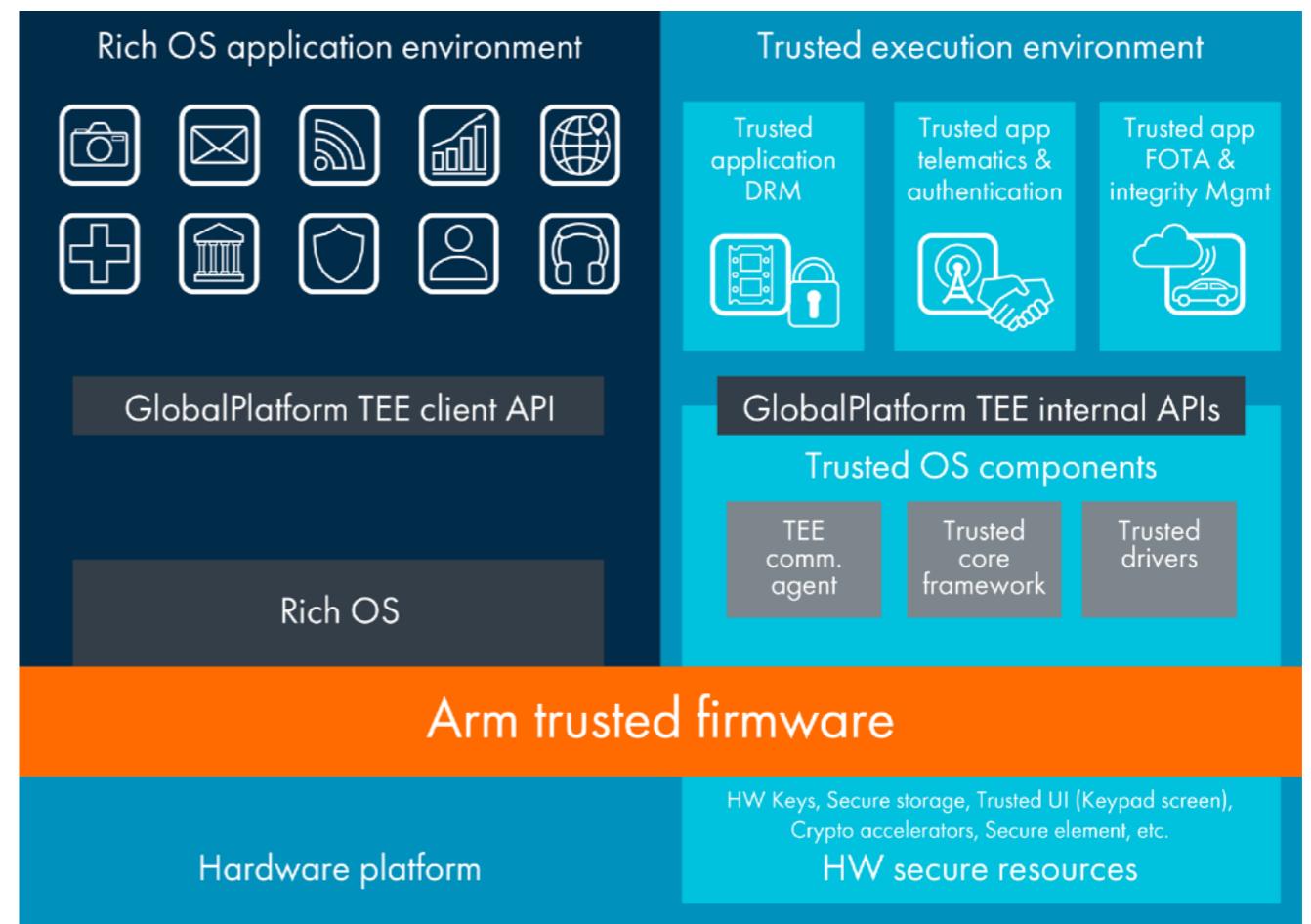


Rust OP-TEE TrustZone SDK

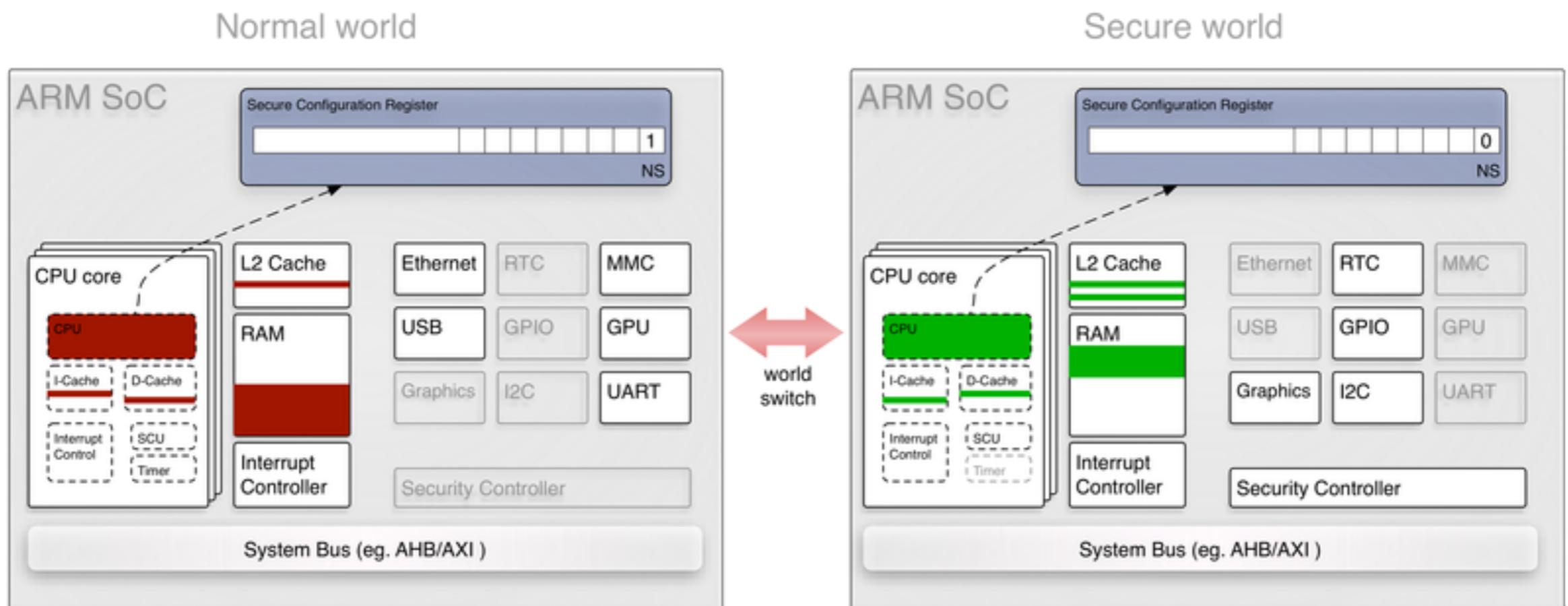
Mingshen Sun
Baidu X-Lab
RustCon Asia, Beijing, April 2019

Background

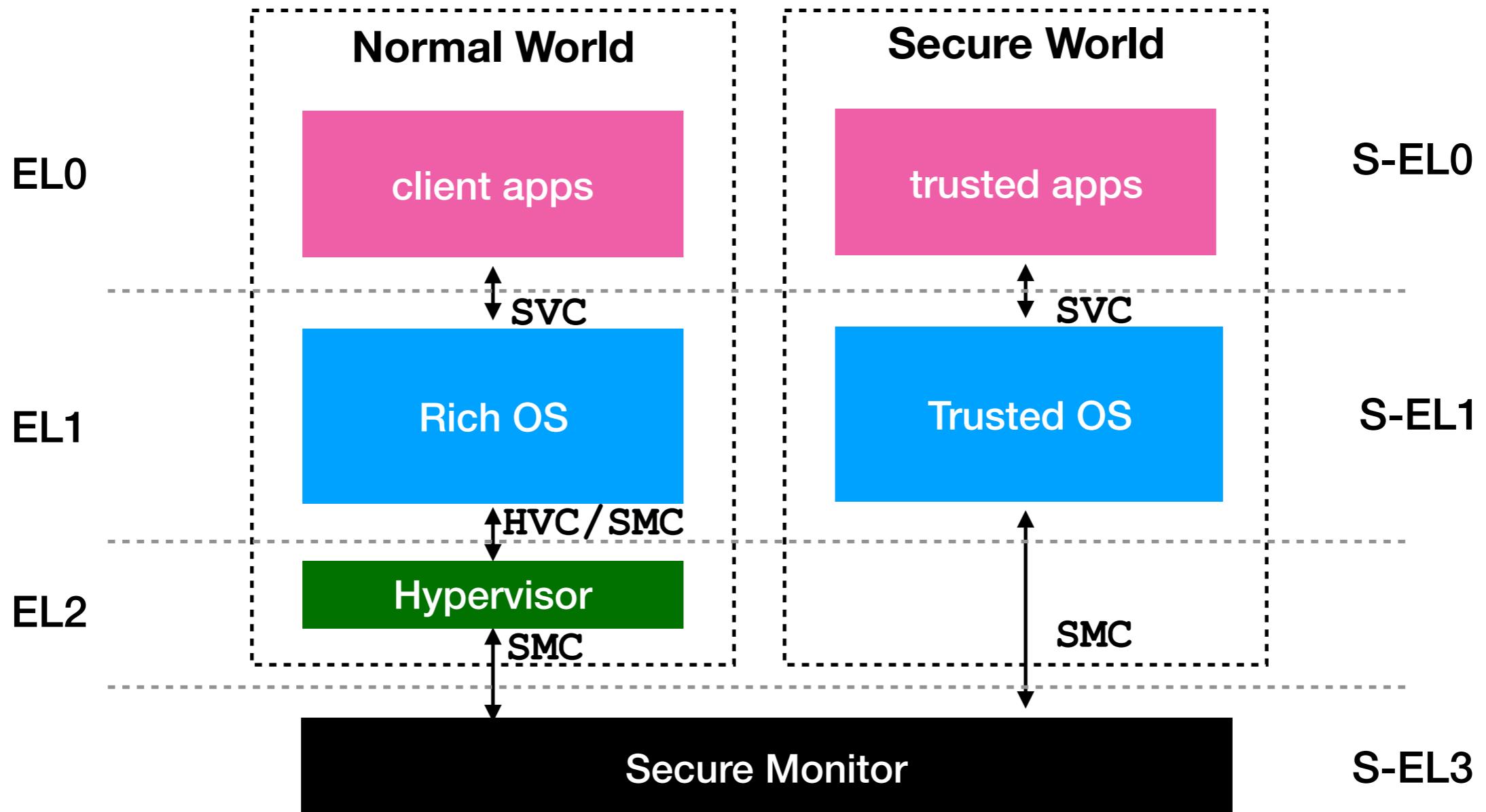
- ARM TrustZone provide ***trusted execution environment*** in mobile phone and embedded devices
- TrustZone secures mobile payment, identification authentication, key management, AI models, DRM, OS integrity, etc.



TrustZone Architecture



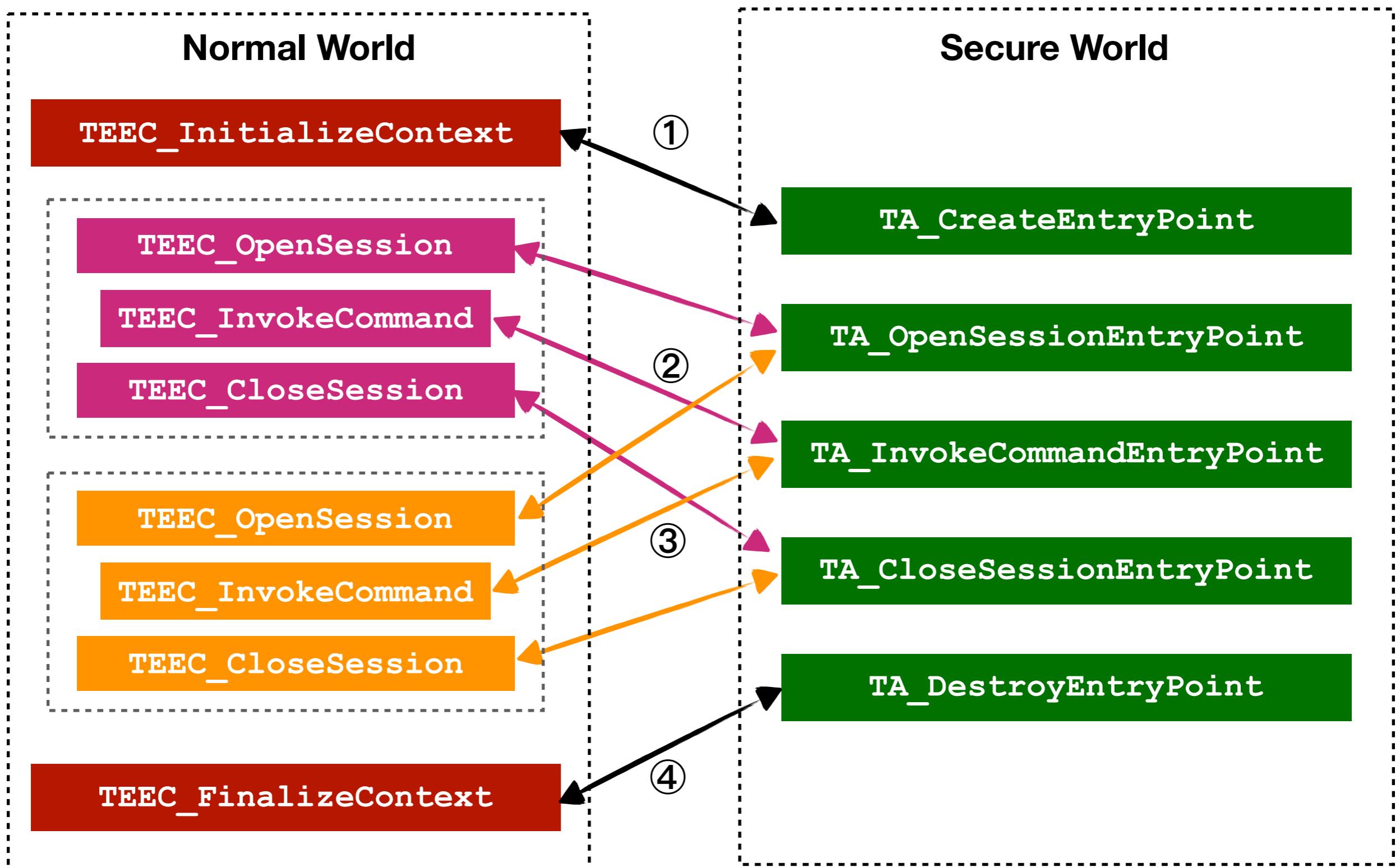
TrustZone Architecture



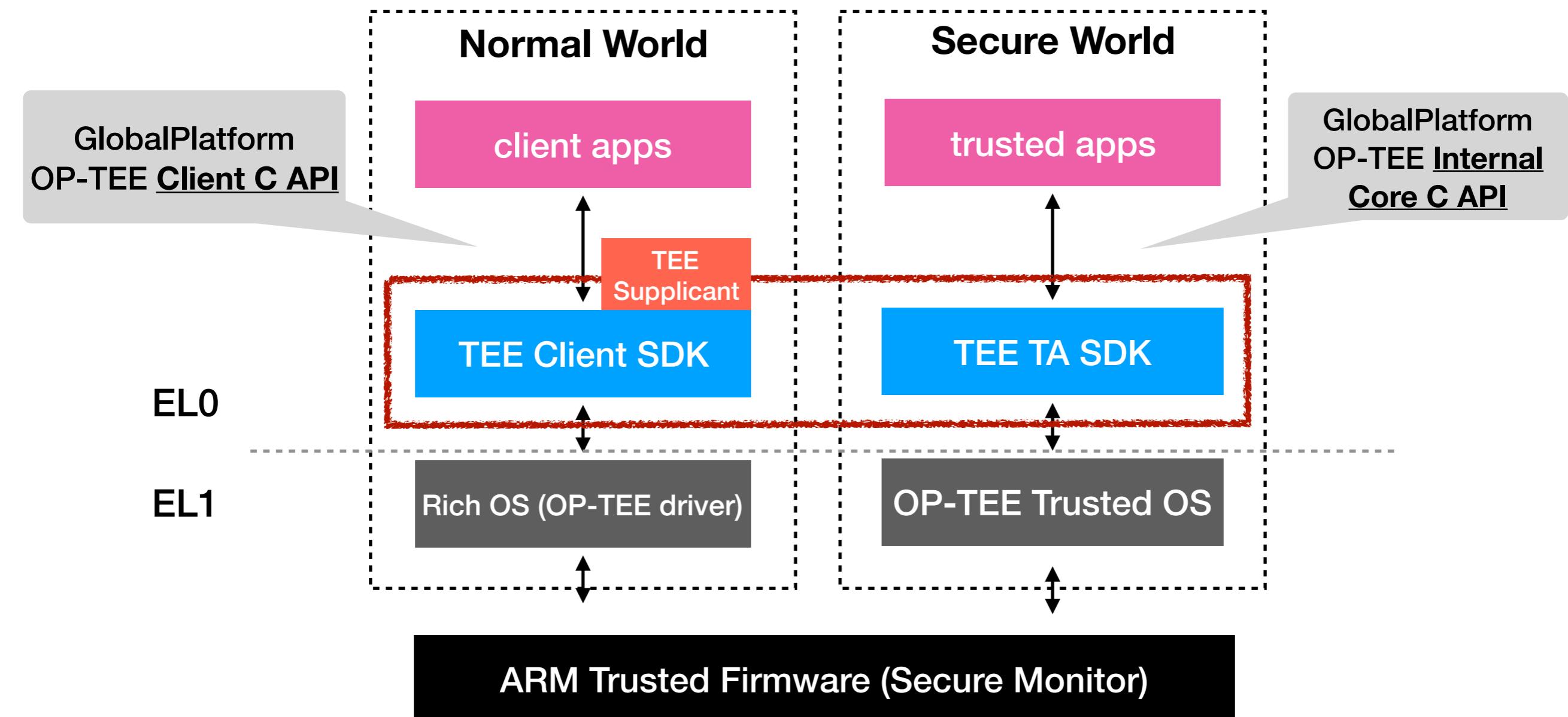
Background

- **GlobalPlatform TEE specifications**
 - *TEE System Architecture (GPD_SPE_009)*: defines a general TEE architecture
 - *TEE Internal Core API Specification (GPD_SPE_010)*
 - *TEE Client API Specification (GPD_SPE_007)*: defines communication interface between Rich OS apps and trusted apps.
- **OP-TEE**: open portable trusted execution environment in compliance with GlobalPlatform specs.

GlobalPlatform TEE API Specification



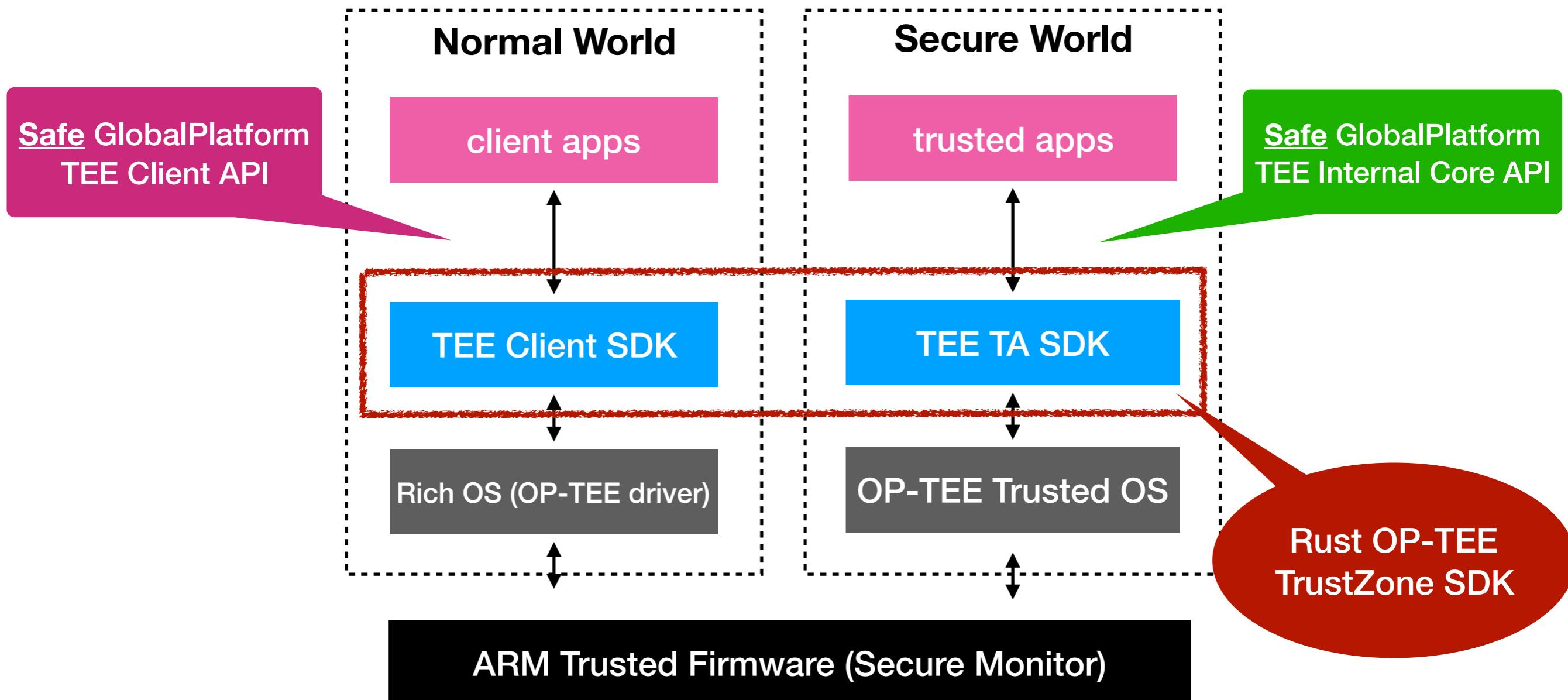
OP-TEE SDK Design



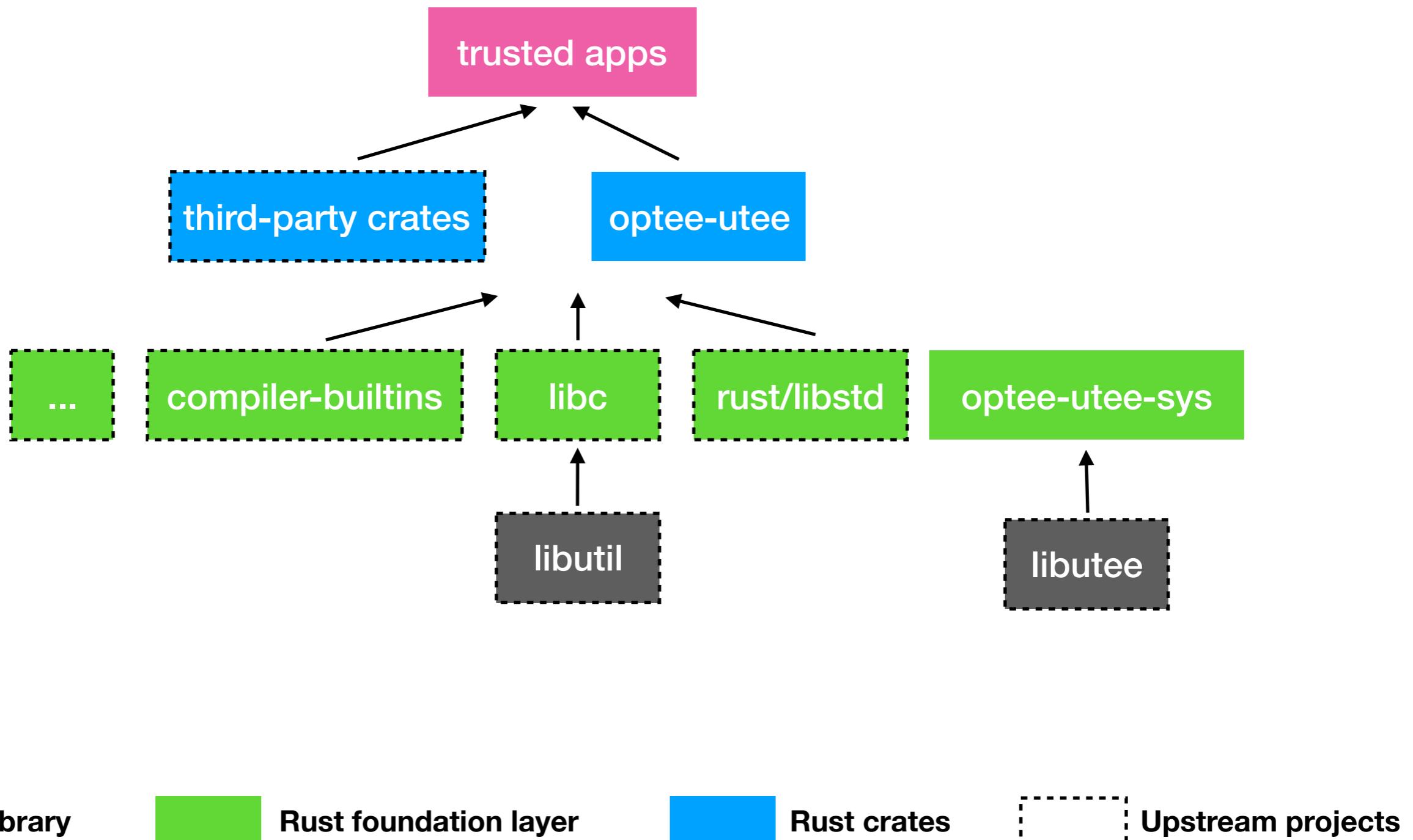
Memory-Safety

- **Memory-safety issues** break security guarantees of TrustZone.
 - Qualcomm's Secure Execution Environment (QSEE) **privilege escalation** vulnerability and exploit (CVE-2015-6639) : <http://bits-please.blogspot.com/2016/05/qsee-privilege-escalation-vulnerability.html>
 - Extracting Qualcomm's KeyMaster Keys - **Breaking Android Full Disk Encryption**: <http://bits-please.blogspot.com/2016/06/extracting-qualcomms-keymaster-keys.html>

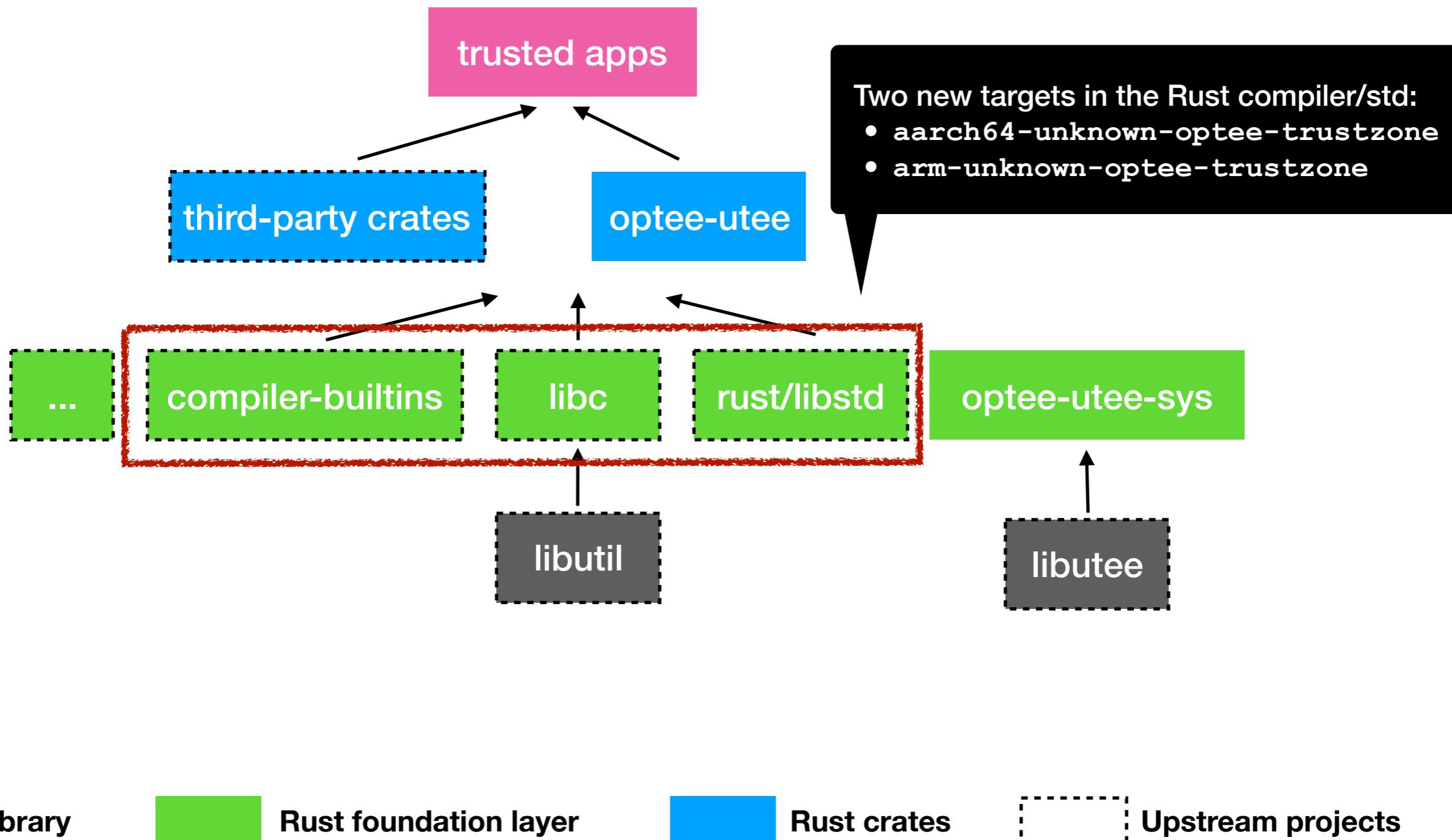
Safe SDK Design



Design of TA SDK



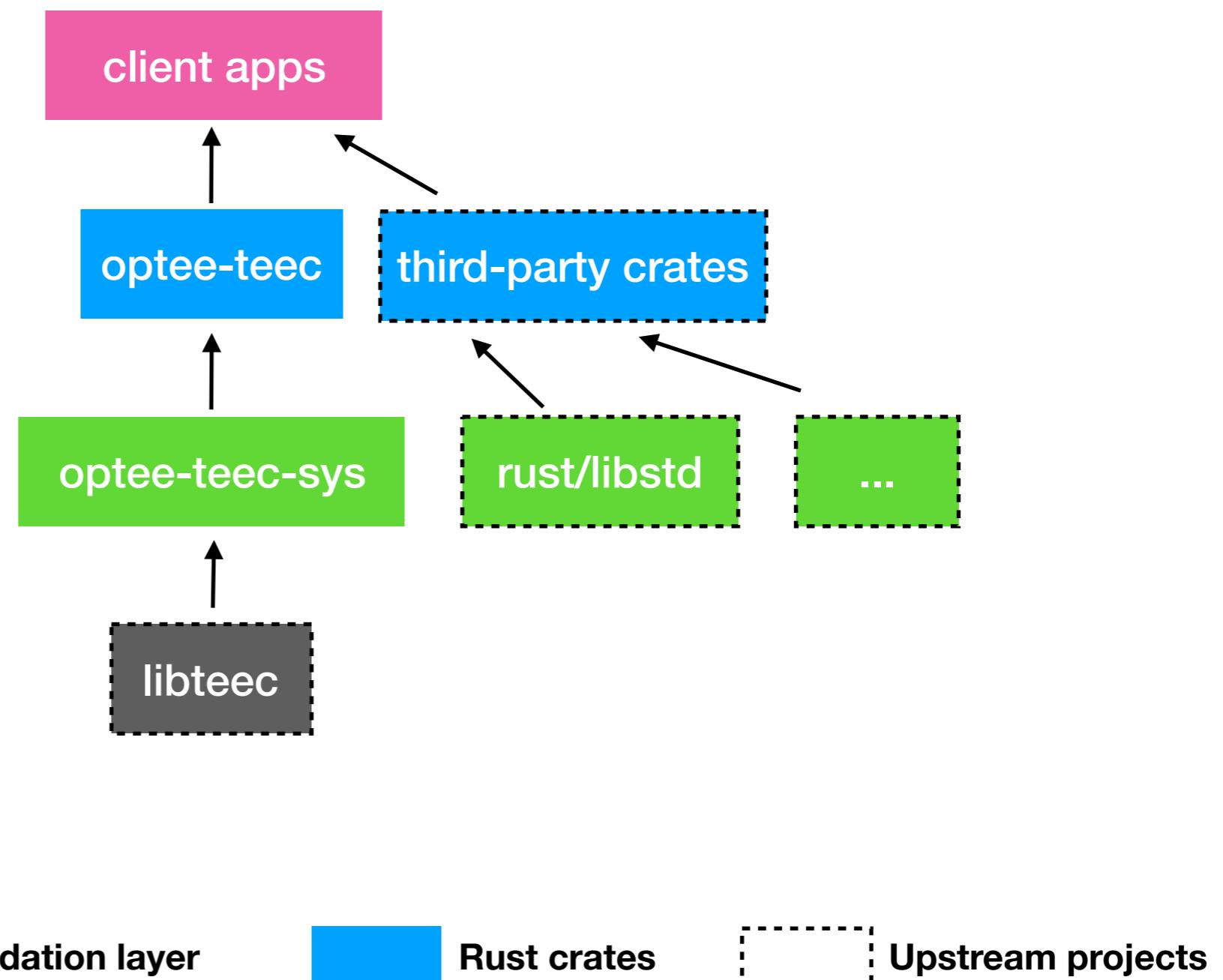
Design of TA SDK



Design of Client SDK

Client apps targets:

- aarch64-unknown-linux-gnu
- arm-unknown-linux-gnu



Project Structure

- **Rust OP-TEE TrustZone SDK:** <https://github.com/mesalock-linux/rust-optee-trustzone-sdk>
- **Rust:** <https://github.com/mesalock-linux/rust>
- **Rust libc:** <https://github.com/mesalock-linux/libc.git>
- **Rust compiler-builtins:** <https://github.com/mesalock-linux/compiler-builtins.git>
- **Wiki:** <https://github.com/mesalock-linux/rust-optee-trustzone-sdk/wiki>

Project Structure

- **optee-teec**: client-side Rust library (LoC: ~933)
- **optee-utee**: TA-side Rust library (LoC: ~2827)
- **optee**: upstream optee library (**optee_client**, **optee_os**)
- **rust**: modified Rust including
 - **rust**: ~29 files changed, 1800 insertions
 - **libc**: ~4 files changed, 131 insertions
 - **compiler-builtins**: ~3 files changed, 3 insertions(+), 1 deletion(-)
- **examples**: **hello_world**, **aes**, **hotp**, **random**, **secure_storage**, and **serde** (LoC: ~3373)

Project Structure - rust/libstd

`src/librustc_target/spec/aarch64_unknown_optee_trustzone.rs`

`src/libstd/sys/optee/alloc.rs`
`src/libstd/sys/optee/args.rs`
`src/libstd/sys/optee/backtrace.rs`
`src/libstd/sys/optee/cmath.rs`
`src/libstd/sys/optee/condvar.rs`
`src/libstd/sys/optee/env.rs`
`src/libstd/sys/optee/fs.rs`
`src/libstd/sys/optee/io.rs`
`src/libstd/sys/optee/memchr.rs`
`src/libstd/sys/optee/mod.rs`
`src/libstd/sys/optee/mutex.rs`

`src/libstd/sys/optee/net.rs`
`src/libstd/sys/optee/os.rs`
`src/libstd/sys/optee/os_str.rs`
`src/libstd/sys/optee/path.rs`
`src/libstd/sys/optee/pipe.rs`
`src/libstd/sys/optee/process.rs`
`src/libstd/sys/optee/rwlock.rs`
`src/libstd/sys/optee/stack_overflow.rs`
`src/libstd/sys/optee/stdio.rs`
`src/libstd/sys/optee/thread.rs`
`src/libstd/sys/optee/thread_local.rs`
`src/libstd/sys/optee/time.rs`

Example: alloc.rs

```
1  use crate::libc;
2  use crate::alloc::{GlobalAlloc, Layout, System};
3
4  #[stable(feature = "alloc_system_type", since = "1.28.0")]
5  unsafe impl GlobalAlloc for System {
6      #[inline]
7      unsafe fn alloc(&self, layout: Layout) -> *mut u8 {
8          libc::malloc(layout.size()) as *mut u8
9      }
10
11     #[inline]
12     unsafe fn alloc_zeroed(&self, layout: Layout) -> *mut u8 {
13         libc::calloc(layout.size(), 1) as *mut u8
14     }
15
16     #[inline]
17     unsafe fn dealloc(&self, ptr: *mut u8, _layout: Layout) {
18         libc::free(ptr as *mut libc::c_void)
19     }
20
21     #[inline]
22     unsafe fn realloc(&self, ptr: *mut u8, _layout: Layout, new_size: usize) -> *mut u8 {
23         libc::realloc(ptr as *mut libc::c_void, new_size) as *mut u8
24     }
25 }
```

The underlying library of libc is
libutil from OP-TEE

Example: thread.rs

```
1 use crate::boxed::FnBox;
2 use crate::ffi::CStr;
3 use crate::io;
4 use crate::sys::{unsupported, Void};
5 use crate::time::Duration;
6
7 pub struct Thread(Void);
8
9 pub const DEFAULT_MIN_STACK_SIZE: usize = 4096;
10
11 impl Thread {
12     // unsafe: see thread::Builder::spawn_unchecked for safety requirements
13     pub unsafe fn new(_stack: usize, _p: Box<dyn FnBox()>)
14         -> io::Result<Thread>
15     {
16         unsupported()
17     }
18
19     pub fn yield_now() {
20         panic!("unsupported")
21     }
22
23     pub fn set_name(_name: &CStr) {
24         panic!("unsupported")
25     }
26
27     pub fn sleep(_dur: Duration) {
28         panic!("unsupported");
29     }
30
31     pub fn join(self) {
32         match self.0 {}
33     }
34 }
35
36 pub mod guard {
37     pub type Guard = !;
38     pub unsafe fn current() -> Option<Guard> { None }
39     pub unsafe fn init() -> Option<Guard> { None }
40 }
```

Thread is not supported in OP-TEE OS. Currently, we will raise a panic.

Getting Started with QEMU

- Rust OP-TEE TrustZone: <https://github.com/mesalock-linux/rust-optee-trustzone-sdk/blob/master/README.md>
- QEMU for ARMv8: <https://github.com/mesalock-linux/rust-optee-trustzone-sdk/wiki/Getting-started-with-OPTEE-for-QEMU-ARMv8>

Getting Started with QEMU

- Clone the project and initialize related submodules

```
$ git clone git@github.com:mesalock-linux/rust-optee-trustzone-sdk.git  
$ cd rust-optee-trustzone-sdk  
$ git submodule update --init  
$ (cd rust/compiler-builtins && git submodule update --init libm compiler-rt)  
$ (cd rust/rust && git submodule update --init src/stdsimd)
```

Getting Started with QEMU

- Clone the project and initialize related submodules
- Install dependencies
- Use Docker instead

```
$ docker build -t rust-optee-trustzone-sdk - < Dockerfile
```

```
$ docker run --rm -it -v$(pwd) :/rust-optee-trustzone-sdk \
-w /rust-optee-trustzone-sdk rust-optee-trustzone-sdk
```

Build Examples

```
$ make optee  
$ source environment  
$ make examples
```

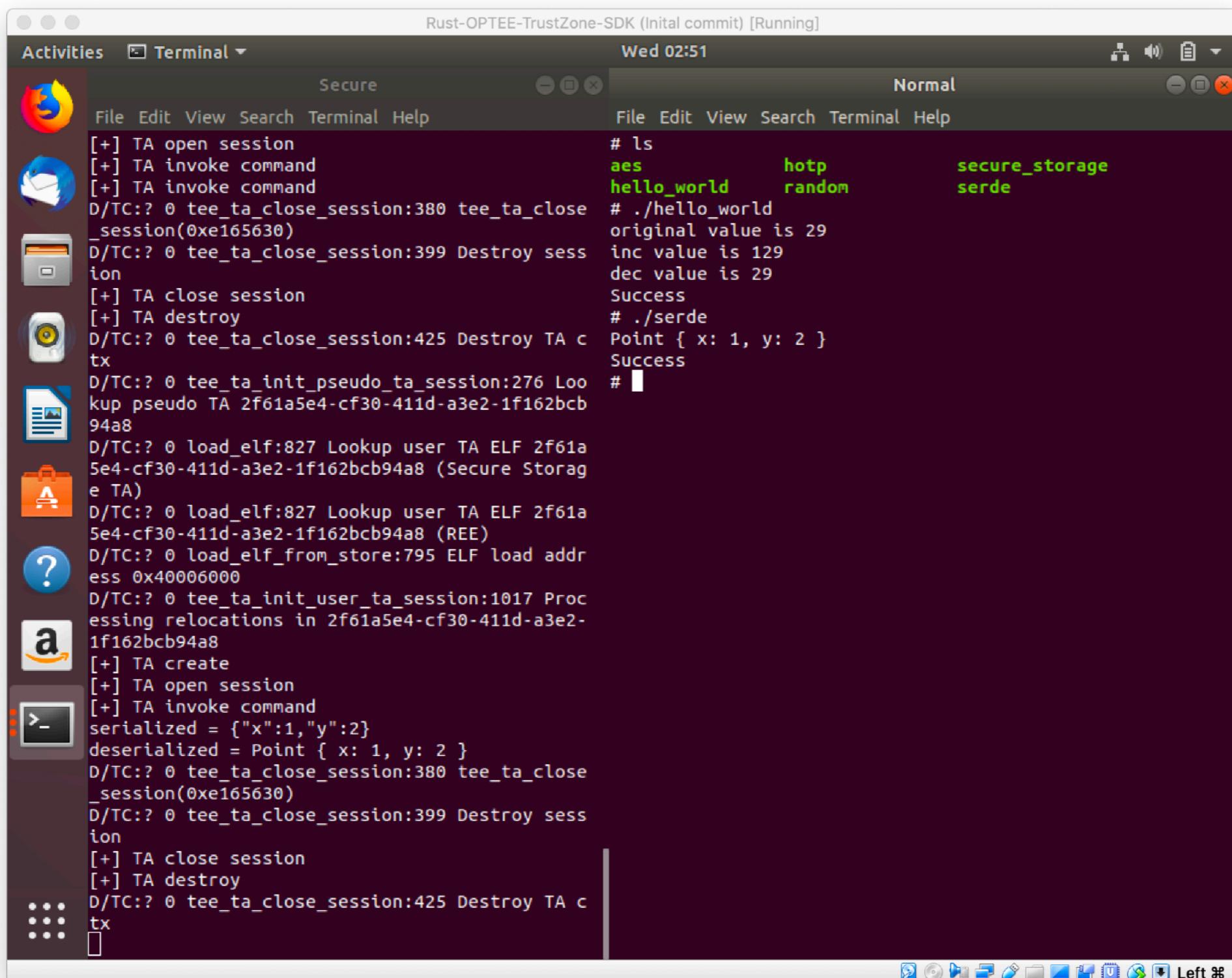
Getting Started with QEMU

- Download OP-TEE for QEMU ARMv8 source code.
- Build OP-TEE for QEMU ARMv8 and images.

```
$ docker run --rm -it -v$(pwd):/rust-optee-trustzone-sdk \
-w /rust-optee-trustzone-sdk \
mesalocklinux/rust-optee-trustzone-sdk-qemuuv8-ci \
bash -c "cd ci && ./ci_hello_world.sh"
```

- Login (username: root) and Mount shared folder in QEMU guest system.
- Copy TAs to corresponding directory.

Example - Demo in QEMU



Rust-OPTEE-TrustZone-SDK (Initial commit) [Running]

Activities Terminal

Secure

```
[+] TA open session
[+] TA invoke command
[+] TA invoke command
D/TC:?: 0 tee_ta_close_session:380 tee_ta_close_session(0xe165630)
D/TC:?: 0 tee_ta_close_session:399 Destroy session
[+] TA close session
[+] TA destroy
D/TC:?: 0 tee_ta_close_session:425 Destroy TA context
D/TC:?: 0 tee_ta_init_pseudo_ta_session:276 Lookup pseudo TA 2f61a5e4-cf30-411d-a3e2-1f162bcb94a8
D/TC:?: 0 load_elf:827 Lookup user TA ELF 2f61a5e4-cf30-411d-a3e2-1f162bcb94a8 (Secure Storage TA)
D/TC:?: 0 load_elf:827 Lookup user TA ELF 2f61a5e4-cf30-411d-a3e2-1f162bcb94a8 (REE)
D/TC:?: 0 load_elf_from_store:795 ELF load address 0x40006000
D/TC:?: 0 tee_ta_init_user_ta_session:1017 Processing relocations in 2f61a5e4-cf30-411d-a3e2-1f162bcb94a8
[+] TA create
[+] TA open session
[+] TA invoke command
serialized = {"x":1,"y":2}
deserialized = Point { x: 1, y: 2 }
D/TC:?: 0 tee_ta_close_session:380 tee_ta_close_session(0xe165630)
D/TC:?: 0 tee_ta_close_session:399 Destroy session
[+] TA close session
[+] TA destroy
D/TC:?: 0 tee_ta_close_session:425 Destroy TA context
```

Normal

```
# ls
aes          hotp
hello_world  random
secure_storage
serde
```

```
# ./hello_world
original value is 29
inc value is 129
dec value is 29
Success
# ./serde
Point { x: 1, y: 2 }
Success
#
```

Example - Client (Initial Design)

```
92 lines (82 sloc) | 2.5 KB
1  #![allow(non_upper_case_globals)]
2  #![allow(non_camel_case_types)]
3  #![allow(non_snake_case)]
4
5  extern crate optee_teecc;
6  pub use libc::*;
7  pub use optee_teecc::*;
8  use std::ptr;
9
10 pub const TA_HELLO_WORLD_CMD_INC_VALUE: u32 = 0;
11 pub const TA_HELLO_WORLD_CMD_DEC_VALUE: u32 = 1;
12
13 pub fn main() {
14     let mut res: TEEC_Result;
15     let mut ctx: TEEC_Context = TEEC_Context {
16         fd: 0,
17         req_memp: true,
18     };
19     let mut sess: TEEC_Session = TEEC_Session {
20         ctx: &mut ctx,
21         session_id: 0,
22     };
23
24     let param1: TEEC_Parameter = TEEC_Parameter {
25         value: TEEC_Value { a: 0, b: 0 },
26     };
27     let param2: TEEC_Parameter = TEEC_Parameter {
28         value: TEEC_Value { a: 0, b: 0 },
29     };
30     let param3: TEEC_Parameter = TEEC_Parameter {
31         value: TEEC_Value { a: 0, b: 0 },
32     };
33     let param4: TEEC_Parameter = TEEC_Parameter {
34         value: TEEC_Value { a: 0, b: 0 },
35     };
36     let param_g: [TEEC_Parameter; 4] = [param1, param2, param3, param4];
37
38     let mut op = TEEC_Operation {
39         started: 0,
40         param_types: 0,
41         params: param_g,
42         session: &mut sess,
43     };
44     let mut err_origin: uint32_t = 0;
45     let mut uuid = TEEC_UUID {
46         time_low: 0x0abcfc200,
47         time_mid: 0x245b,
48         time_hi_and_version: 0x1le4,
49         clock_seq_and_node: [0xab, 0xe2, 0x00, 0x02, 0xa5, 0xd5, 0xc5, 0x1b],
50     };
51
52     unsafe {
53         res = TEEC_InitializeContext(ptr::null_mut() as *mut c_char, &mut ctx);
54         if res != TEEC_SUCCESS {
55             println!("Init error.\n");
56             return;
57         }
58
59         res = TEEC_OpenSession(
60             &mut ctx,
61             &mut sess,
62             &mut uuid,
63             TEEC_LOGIN_PUBLIC,
64             ptr::null() as *const c_void,
65             ptr::null_mut() as *mut TEEC_Operation,
66             &mut err_origin,
67         );
68         if res != TEEC_SUCCESS {
69             println!("Open session error.\n");
70             return;
71         }
72
73         op.param_types = TEEC_PARAM_TYPES(TEEC_VALUE_INOUT, TEEC_NONE, TEEC_NONE, TEEC_NONE);
74         op.params[0].value.a = 29;
75         println!("original value is {}", op.params[0].value.a);
76         res = TEEC_InvokeCommand(
77             &mut sess,
78             TA_HELLO_WORLD_CMD_INC_VALUE,
79             &mut op,
80             &mut err_origin,
81         );
82         if res != TEEC_SUCCESS {
83             println!("Execute command error.\n");
84             return;
85         }
86         println!("update value is {}", op.params[0].value.a);
87
88         TEEC_CloseSession(&mut sess);
89     }
90 }
```

```
raw::TEEC_Context  
raw::TEEC_Session  
raw::TEEC_Parameter  
raw::TEEC_Operation  
raw::TEEC_InitializeContext  
raw::TEEC_OpenSession  
raw::TEEC_InvokeCommand  
raw::TEEC_CloseSession  
raw::TEEC_FinalizeContext
```

```
unsafe {  
}  
}
```

Example - Client (Current Design)

```
1 use optee_teecc::Context, Operation, ParamType, Session, Uuid>;
2 use optee_teecc::ParamNone, ParamValue;
3 use proto::self, Command;
4
5 fn hello_world(session: &mut Session) -> optee_teecc::Result<()> {
6     let p0 = ParamValue::new(29, 0, ParamType::ValueInout);
7     let mut operation = Operation::new(0, p0, ParamNone, ParamNone, ParamNone);
8
9     println!("original value is {:?}", operation.parameters().0.a());
10
11    session.invoke_command(Command::IncValue as u32, &mut operation)?;
12    println!("inc value is {:?}", operation.parameters().0.a());
13
14    session.invoke_command(Command::DecValue as u32, &mut operation)?;
15    println!("dec value is {:?}", operation.parameters().0.a());
16    Ok(())
17}
18
19 fn main() -> optee_teecc::Result<()> {
20     let mut ctx = Context::new()?;
21     let uuid = Uuid::parse_str(proto::UUID).unwrap();
22     let mut session = ctx.open_session(uuid)?;
23
24     hello_world(&mut session)?;
25
26     println!("Success");
27     Ok(())
28 }
```

Annotations pointing to specific code snippets:

- Annotation: `ParamValue::new()` points to the line `let p0 = ParamValue::new(29, 0, ParamType::ValueInout);`
- Annotation: `Operation::new()` points to the line `let mut operation = Operation::new(0, p0, ParamNone, ParamNone, ParamNone);`
- Annotation: `session.invoke_command()` points to the line `session.invoke_command(Command::IncValue as u32, &mut operation)?;`
- Annotation: `Context::new()` points to the line `let mut ctx = Context::new()?;`
- Annotation: `ctx.open_session()` points to the line `let mut session = ctx.open_session(uuid)?;`

Example - Trusted App (First Commit)

```
1  #![allow(non_upper_case_globals)]
2  #![allow(non_camel_case_types)]
3  #![allow(non_snake_case)]
4
5  extern crate optee_utee;
6  pub use optee_utee::*;

7
8  #[no_mangle]
9  pub extern "C" fn TA_CreateEntryPoint() -> TEE_Result {
10     return TEE_SUCCESS;
11 }
12
13 #[no_mangle]
14 pub extern "C" fn TA_DestroyEntryPoint() {
15 }
16
17 #[no_mangle]
18 pub extern "C" fn TA_OpenSessionEntryPoint(_paramTypes: ParamTypes, _params: TEE_Param, _sessionContext: SessionP) -> TEE_Result {
19     return TEE_SUCCESS;
20 }
21
22 #[no_mangle]
23 pub extern "C" fn TA_CloseSessionEntryPoint(_sessionContext: SessionP) {
24 }
25
26 #[no_mangle]
27 pub extern "C" fn TA_InvokeCommandEntryPoint(_sessionContext: SessionP, _commandID: u32) -> TEE_Result {
28     match _commandID {
29         0 => {
30             unsafe { _params[0].value.a += 121; }
31         },
32         1 => {
33             unsafe { _params[0].value.a -= 21; }
34         },
35         _ => {
36             return TEE_ERROR_BAD_PARAMETERS;
37         }
38     }
39     return TEE_SUCCESS;
40 }
```

```
# [no_mangle]

pub extern "C" fn TA_CreateEntryPoint() -> TEE_Result {
    return TEE_SUCCESS;
}
```

```
# [no_mangle]

pub extern "C" fn TA_OpenSessionEntryPoint(
    _paramTypes: ParamTypes,
    _params: TEE_Param,
    _sessionContext: SessionP) -> TEE_Result {
    return TEE_SUCCESS;
}
```

```
0 => {

    unsafe { _params[0].value.a += 121; }

},
```

Example - Trusted App (Current Design)

```
8
9 #[ta_create]
10 fn create() -> Result<()> {
11     trace_println!("[+] TA create");
12     Ok(())
13 }
14
15 #[ta_open_session]
16 fn open_session(_params: &mut Parameters) -> Result<()> {
17     trace_println!("[+] TA open session");
18     Ok(())
19 }
20
21 #[ta_close_session]
22 fn close_session() {
23     trace_println!("[+] TA close session");
24 }
25
26 #[ta_destroy]
27 fn destroy() {
28     trace_println!("[+] TA destroy");
29 }
30
31 #[ta_invoke_command]
32 fn invoke_command(cmd_id: u32, params: &mut Parameters) -> Result<()> {
33     trace_println!("[+] TA invoke command");
34     let mut values = unsafe { params.0.as_value().unwrap() };
35     match Command::from(cmd_id) {
36         Command::IncValue => {
37             values.set_a(values.a() + 100);
38             Ok(())
39         }
40         Command::DecValue => {
41             values.set_a(values.a() - 100);
42             Ok(())
43         }
44         _ => Err(Error::new(ErrorKind::BadParameters)),
45     }
46 }
```

[ta_create]

[ta_open_session]

[ta_close_session]

[ta_destory]

[ta_invoke_command]

Example - Trusted App (Current Design)

```
48 // TA configurations
49 const TA_FLAGS: u32 = 0;
50 const TA_DATA_SIZE: u32 = 32 * 1024;
51 const TA_STACK_SIZE: u32 = 2 * 1024;
52 const TA_VERSION: &[u8] = b"0.1\0";
53 const TA_DESCRIPTION: &[u8] = b"This is a hello world ex
54 const EXT_PROP_VALUE_1: &[u8] = b"Hello World TA\0";
55 const EXT_PROP_VALUE_2: u32 = 0x0010;
56 const TRACE_LEVEL: i32 = 4;
57 const TRACE_EXT_PREFIX: &[u8] = b"TA\0";
58 const TA_FRAMEWORK_STACK_SIZE: u32 = 2048;
59
60 include!(concat!(env!("OUT_DIR"), "/user_ta_header.rs"));
```

TA configurations

TA_DATA_SIZE: heap size
TA_STACK_SIZE: stack size

Include some static data structures

Example - Project Structure

- **host/**: source code of the client app
- **ta/**: source code of TA
 - **ta.lds**: linker script
 - **xargo.toml**: "Cargo.toml" for cross compilation
 - **ta_static.rs**: some static data structure for TA
- **proto/**: shared data structure and configurations like a protocol
- **Makefile**: Makefile to build host and client
- **uuid.txt**: UUID for TA, randomly generated if the file does not exist.

arm UNKNOWN linux gnu
aarch64 UNKNOWN linux gnu

arm UNKNOWN optee trustzone
aarch64 UNKNOWN optee trustzone

Example - Use Serde

```
1 [package]
2 name = "ta"
3 version = "0.1.0"
4 authors = ["The Rust OP-TEE TrustZone SDK Project Developers"]
5 license = "Apache-2.0"
6 repository = "https://github.com/mesalock-linux/rust-optee-trustzone-sdk.git"
7 description = "An example of Rust OP-TEE TrustZone SDK."
8 edition = "2018"
9
10 [dependencies]
11 libc = { path = "../../rust/libc" }
12 optee-utee-sys = { path = "../../optee-utee/optee-utee-sys" }
13 optee-utee = { path = "../../optee-utee" }
14 serde = { version = "1.0", features = ["derive"] }
15 serde_json = "1.0"
16
17 [build_dependencies]
18 uuid = { version = "0.7", features = ["v4"] }
```

serde / serde_json

Example - Use Serde

```
37 #[ta_invoke_command]
38 fn invoke_command(cmd_id: u32, _params: &mut Parameters) -> Result<()> {
39     trace_println!("[+] TA invoke command");
40     match Command::from(cmd_id) {
41         Command::DefaultOp => {
42             let point = Point { x: 1, y: 2 };
43
44             // Convert the Point to a JSON string.
45             let serialized = serde_json::to_string(&point).unwrap();
46
47             // Prints serialized = {"x":1,"y":2}
48             trace_println!("serialized = {}", serialized);
49
50             // Convert the JSON string back to a Point.
51             let deserialized: Point = serde_json::from_str(&serialized).unwrap();
52
53             // Prints deserialized = Point { x: 1, y: 2 }
54             trace_println!("deserialized = {:?}", deserialized);
55
56             Ok(())
57         }
58         - => Err(Error::new(ErrorKind::BadParameters)),
59     }
60 }
```

Use `serde` to handle invoke command

Trusted Storage API for Data and Keys

- Trusted Storage Spaces
 - Transient objects
 - Persistent objects

Trusted Storage API for Data and Keys

Transient Objects

Figure 5-1: State Diagram for TEE_ObjectHandle



Persistent Objects

Other Examples

- **hello_world**: minimal project structure
- **aes**: crypto, shared memory APIs
- **hotp**: crypto APIs
- **random**: crypto APIs
- **secure_storage**: secure object related APIs
- **serde**: Rust third-party crates for de/serialization

Documents & Wiki

Struct optee_tee::Context

[+] Show declaration
[-] An abstraction of the logical connection between a client application and a TEE.

Methods

- [-] `impl Context`
- [-] `pub fn new() -> Result<Context>`
Creates a TEE client context object.
- Examples**

```
let ctx = Context::new().unwrap();
```

- [-] `pub fn new_raw(fd: c_int, reg_mem: bool) -> Result<TEC_Context>`
Creates a raw TEE client context with implementation defined parameters.
- Examples**

```
let raw_ctx: optee_tee::sys::TEEC_Context = Context::new_raw(0, true);
```

- [-] `pub fn as_mut_raw_ptr(&mut self) -> *mut TEEC_Context`
Converts a TEE client context to a raw pointer.
- Examples**

```
let mut ctx = Context::new().unwrap();
let mut raw_ptr: *mut optee_tee::sys::TEEC_Context = ctx.as_mut_raw_ptr();
```

- [-] `pub fn open_session(&mut self, uuid: Uuid) -> Result<Session>`
Opens a new session with the specified trusted application.
The target trusted application is specified by `uuid`.
- Examples**

```
let mut ctx = Context::new().unwrap();
let uuid = Uuid::parse_str("8abcf200-2450-11e4-abe2-0002a5d5c51b");
let session = ctx.open_session(uuid).unwrap();
```

- [-] `pub fn open_session_with_operation(&mut self, uuid: Uuid, operation: Operation) -> Result<Session>`
Opens a new session with the specified trusted application, pass some parameters to TA by a parameter object.
The target trusted application is specified by `uuid`.
- Examples**

```
let mut ctx = Context::new().unwrap();
let uuid = Uuid::parse_str("8abcf200-2450-11e4-abe2-0002a5d5c51b");
let p0 = Parameter::from_value(42, 0, ParamType::ValueInout);
let p1 = Parameter::new();
let p2 = Parameter::new();
let p3 = Parameter::new();
let mut operation = Operation::new(0, p0, p1, p2, p3);
let session = ctx.open_session_with_operation(uuid, operation).unwrap();
```

Trait Implementations

 - [+] `impl Drop for Context`

Auto Trait Implementations

Enum optee_tee::ErrorKind

[+] Show declaration
[-] A list specifying general categories of TEE client error and its corresponding code in OP-TEE client library.

Variants

- Generic**
[-] Non-specific cause.
- AccessDenied**
[-] Access privileges are not sufficient.
- Cancel**
[-] The operation was canceled.
- AccessConflict**
[-] Concurrent accesses caused conflict.
- ExcessData**
[-] Too much data for the requested operation was passed.
- BadFormat**
[-] Input data was of invalid format.
- BadParameters**
[-] Input parameters were invalid.
- BadState**
[-] Operation is not valid in the current state.
- ItemNotFound**
[-] The requested data item is not found.
- NotImplemented**
[-] The requested operation should exist but is not yet implemented.
- NotSupported**
[-] The requested operation is valid but is not supported in this implementation.
- NoData**
[-] Expected data was missing.
- OutOfMemory**
[-] System ran out of resources.
- Busy**
[-] The system is busy working on something else.
- Communication**
[-] Communication with a remote party failed.
- Security**
[-] A security fault was detected.
- ShortBuffer**
[-] The supplied buffer is too short for the generated output.
- ExternalCancel**
[-] Implementation defined error code.
- TargetDead**
[-] Implementation defined error code: trusted Application has panicked during the operation.
- Unknown**
[-] Unknown error.

Trait Implementations

- [+] `impl Clone for ErrorKind`
- [+] `impl Ord for ErrorKind`
- [+] `impl From<ErrorKind> for Error`
- `impl Eq for ErrorKind`
- `impl Copy for ErrorKind`
- [+] `impl PartialOrd<ErrorKind> for ErrorKind`
- [+] `impl PartialEq<ErrorKind> for ErrorKind`
- [+] `impl Hash for ErrorKind`

Getting started with OPTEE for QEMU ARMv8

Mingshen Sun edited this page 21 hours ago · 2 revisions

To run examples on the QEMU ARMv8 emulator, we need first build OP-TEE for QEMU ARMv8. You can install dependencies with this [instruction](#) or use our [Dockerfile](#).

Download OP-TEE for QEMU ARMv8 source code.

```
$ mkdir -p ~/bin
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo && chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
$ mkdir optee-qemuuv0-3.4.0 && cd optee-qemuuv0-3.4.0 && \
repo init -q -u https://github.com/OP-TEE/manifest.git -m qemu_v8.xml -b 3.4.0 && \
repo sync -j4 --no-clone-bundle
```

Build OP-TEE for QEMU ARMv8 and images.

```
$ cd build && \
make -j2 toolchains && \
make QEMU_VIRTFS_ENABLE=y CFG_TEE_RAM_VA_SIZE=0x00300000 -j5(nproc)
```

Create a shared folder to share example host apps and TAs with QEMU guest system.

```
$ mkdir shared_folder
$ cp ALL_HOST_APPS_TAS shared_folder/
```

Run QEMU.

```
$ make run-only QEMU_VIRTFS_ENABLE=y QEMU_VIRTFS_HOST_DIR=$(pwd)/shared_folder
```

Mount shared folder in QEMU guest system (username: root).

```
$ mkdir shared && mount -t 9p -o trans=virtio host shared
```

Copy TAs to corresponding directory.

```
$ cd shared && cp *.ta /lib/optee_armitz/
```

Execute host apps.

```
$ ./hello_world
original value is 29
inc value is 129
dec value is 29
Success
```

Note that if you are under a environment without GUI, you please comment out the following code in `qemu_v8.mk` and use `nc` instead.

```
diff --git a/qemu_v8.mk b/qemu_v8.mk
index 8271590..1c4a91b 100644
--- a/qemu_v8.mk
+++ b/qemu_v8.mk
@@ -163,9 +163,9 @@ run-only:
    ln -sf $(ROOT)/out-br/images/rootfs.cpio.gz $(BINARIES_PATH)/
    $(call check-terminal)
    $(call run-help)
-   $(call launch-terminal,54320,"Normal World")
-   $(call launch-terminal,54321,"Secure World")
-   $(call wait-for-ports,54320,54321)
+   # $(call launch-terminal,54320,"Normal World")
+   # $(call launch-terminal,54321,"Secure World")
+   # $(call wait-for-ports,54320,54321)
    cd $(BINARIES_PATH) && $(QEMU_PATH)/aarch64-softmmu/qemu-system-aarch64 \
        -nographic \
        -serial tcp:localhost:54320 -serial tcp:localhost:54321
```

Before start QEMU, run two `nc` to listen port 54320 and 54321.

```
$ nc -l 127.0.0.1 54320
$ nc -l 127.0.0.1 54321
```

Then open QEMU by `make run-only`, and start by input `c`.

Test

- **ctest**: Automated testing of FFI bindings in Rust. This repository is intended to validate the `*-sys` crates that can be found on crates.io to ensure that the APIs in Rust match the APIs defined in C.
- QEMU integration tests, no unit tests for now
- Travis CI

Run/Test Examples in QEMU

```
$ docker run --rm -it \
-v$(pwd):/rust-optee-trustzone-sdk \
-w /rust-optee-trustzone-sdk \
mesalocklinux/rust-optee-trustzone-sdk-qemuuv8-ci \
bash -c "cd ci && ./ci_hello_world.sh"
```

Roadmap

- **April:** open source
- **May:** trusted storage API design, cryptographic operations API design, TEE arithmetical API design, and more third-party Rust crates
- **Jun:** push modified Rust compiler/std to upstream and make OP-TEE TrustZone as an official target.
- **2019 Q3/4:** more trusted apps such as secure key service, remote attestation, fTPM, and machine learning algorithm.

Summary

- TrustZone 为手机、嵌入式设备提供安全的可信执行环境，用于包括安全支付、密钥管理、模型保护等场景。但是由于内存安全问题，TrustZone 中运行的安全应用 (trusted application, TA) 的安全性大打折扣。
- Rust OP-TEE TrustZone SDK 为当今广泛使用的开源 TrustZone 实现 OP-TEE 提供了一套内存安全、使用方便的 SDK。SDK 基于 GlobalPlatform 的 TEE 标准，为开发者提供标准的开发接口。除此之外 Rust OP-TEE TrustZone SDK 支持标准库和第三方库，提高了 TA 的开发速度，并扩展了 TrustZone 的应用场景。同时，它是 MesaTEE 项目的一部分。
- **License:** Apache v2