# Open-TEE - An Open Virtual Trusted Execution Environment

Peng Xu

November 7, 2018

## Table of Contents

## Introduction

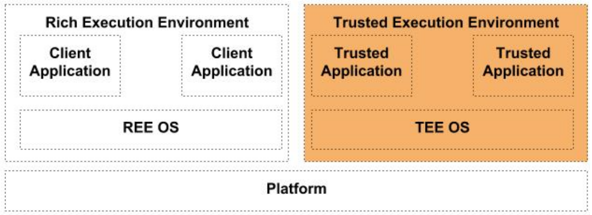
- Why we need hardware-based TEEs?
- TEEs are programmable (TPMs/HSMs)
- Application developers have lacked the interfaces to use hardware-based TEE functionality
- Software development kits are proprietary or expensive

## Introduction

- Why we need hardware-based TEEs?
- TEEs are programmable (TPMs/HSMs)
- Application developers have lacked the interfaces to use hardware-based TEE functionality
- Software development kits are proprietary or expensive
- Open-tee
  1. Not intended to emulate a hardware TEE
  2. Compile and run Trusted Application successfully on any TEE-compliant targets

## Background - Structure

- Rich Execution Environment (REE)
- Trusted Execution Environment (TEE)
- Trusted Application (TA)
- Client Application (CA)



## TEE architectural options

- Co-Processor
  - External Security co-processor: outside of main System on Chip (SoC)
  - Embedded Security co-processor: embedded into the main SoC

## TEE architectural options

- Co-Processor
  - External Security co-processor: outside of main System on Chip (SoC)
  - Embedded Security co-processor: embedded into the main SoC
- Processor Secure Environment

## TEE architectural options

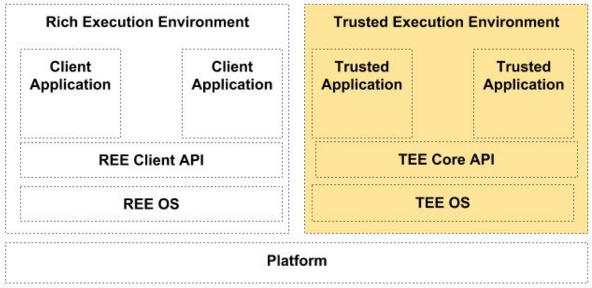
- Processor Secure Environment
    - ARM TrustZone
    - Intel Software Guard Extensions (SGX)

## Why Open-TEE?

1. Enable to utilize TEE functionality
2. Provide a fast and efficient prototyping environment
3. Promote research into TEE Services
4. Promote community involvement

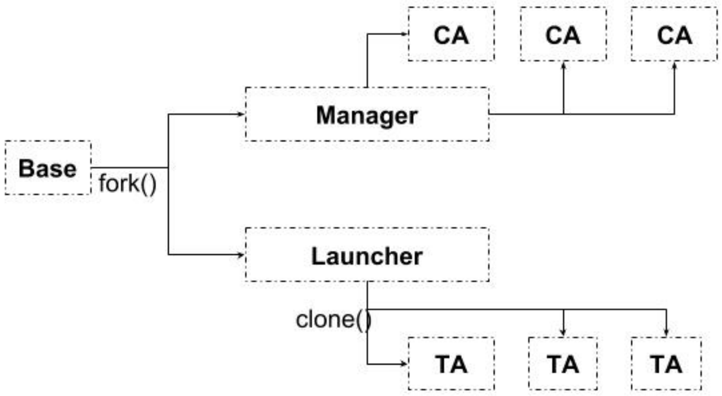
## Architecture of Open-TEE

1. REE Client API and TEE Core API



2. Requirements
   2.1 Compliance and ease-of-use
   2.2 Hardware-independence
   2.3 Reasonable Performance

## Architecture of Open-TEE



## Architecture of Open-TEE - Base

1. A process that encapsulates the TEE functionality as a whole
2. Loading the configuration
3. Preparing the common parts of the system
4. Forking two processes: Manager and Launcher

## Architecture of Open-TEE - Manager

1. Open-TEE's operating system
2. Manager's responsibilities:
   - 2.1 Managing connections between applications
   - 2.2 Monitoring TA state
   - 2.3 Providing secure storage for a TA
   - 2.4 Controlling shared memory regions for the connected application

## Architecture of Open-TEE - Launcher

1. Creating new TA processes
2. Loading TEE Core API library
3. Waiting commands from Manager

## Architecture of Open-TEE - TA Processes

1. Each process is divided into two threads
2. Inter-process Communication (IPC) thread
3. TA logic thread

## Evaluation

Questions?