1 Open-TEE - An Open Virtual Trusted Execution Environment

1.1 Introduction

The following contributions are made:

- Design and implementation of a virtual TEE called Open-TEE which conforms to GlobalPlatform Specifications.
- It is shown that Open-TEE is efficient, hardware-independent and allows a developer to carry out much of the development life cycle of standard-compliant TEE applications using popular application development environments they already use.

1.2 Background

One initiative in TEE standardization has been undertaken by GlobalPlatform (GP), which is a cross industry, non-profit association which identifies, develops and publishes specifications that promote the secure and interoperable deployment and management of multiple applications on secure chip technology. These standardization efforts in GlobalPlatform could resolve the issue of inter-operable TEEs. However, they do not remove the obstacle in gaining access to the requisite hardware nor does simplify the task of developing and testing TAs.

1.3 OPen-TEE

The motivations are

- Enable developer access to TEE functionality
- Provide a fast and efficient prototyping environment
- Promote research into TEE Services
- Promote community involvement

The requirements are

- Compliance: Our framework should comply with GP's main interfaces, the Client and Core APIs.
- Hardware-independence: As a software based solution the framework should not be dependent on a particular TEE hardware environment, neither should the development system itself.
- Reasonable Performance: The framework must minimize the on-disk footprint and the memory consumption required to run it, the start-up and restart times should also be reasonable.
- Ease-of-use: The solution should be easily deployed and configured.

Architecture The architecture is split up in the following parts

- Base: Open-TEE is designed to function as a daemon process in user space. It starts executing Base, a process that encapsulates the TEE functionality as a whole. Base is responsible for loading the configuration and preparing the common parts of the system. Once initialized the Base will fork to create two independent but related processes. One process becomes Manager and the other, Launcher which serves as a prototype for TAs.

- Manager: Manager can be visualized as Open-TEE's "operating system". Its main responsibilities are: managing connections between applications, monitoring TA state, providing secure storage for a TA and controlling shared memory regions for the connected applications.

- Launcher: The sole purpose of Launcher is to create new TA processes efficiently. When it is first created, Launcher will load a shared library implementing the TEE Core API and will wait for further commands from Manager. Manager will signal Launcher when there is a need to launch a new TA. Upon receiving the signal, Launcher will clone itself. The clone will then load the shared library corresponding to the requested TA.

- Trusted Application Processes: The architecture of the TA processes is inspired by the multi-process architecture. Each process has been divided into two threads. The first handles Inter-Process Communication (IPC) and the second is the working thread, referred to respectively as the IO and TA Logic threads.

1.4 Evaluation

By following the GP standard and not emulating any specific TEE hardware, Open-TEE is independent of TEE hardware. TAs developed with Open-TEE can be compiled to any target TEE hardware architecture. We have verified that a non-trivial TA developed using Open-TEE has been succesfully compiled and run on a hardware TEE based on ARM TrustZone.