

# MRTM Software Implementation for Mobile Secure Environments

Sven Bugiel\*

Royal Institute of Technology / Technical University of Denmark

bugiel@kth.se

## Abstract

On this poster we present a software-based implementation of a Mobile Remote Owner Trusted Module, using security extensions of contemporary System-on-Chip architectures. An explicit challenge are the constrained resources of such on-chip mechanisms. We expose a software architecture that minimizes the code and data size of the module, applying some novel approaches proposed in recent research.

## Introduction

A Mobile Trusted Module (MTM) is conceptually a Trusted Platform Module (TPMv.12) as defined in the Trusted Computing Group specifications. However, it differs in some aspects from the TPM design in order to meet the requirements of mobile and embedded devices. For example, MTM mandates only a minor subset of the TPM commands, but includes extra commands and control specifications needed for mobile phone specific use cases (e.g. secure boot, RIMs).

Moreover, the specifications define two interleaving profiles for a MTM, depending on the owner entity – the Mobile Local Owner Trusted Module (MLTM), i.e. the user, and the Mobile Remote Owner Trusted Module (MRTM), e.g. a device manufacturer or

## Target Adaption

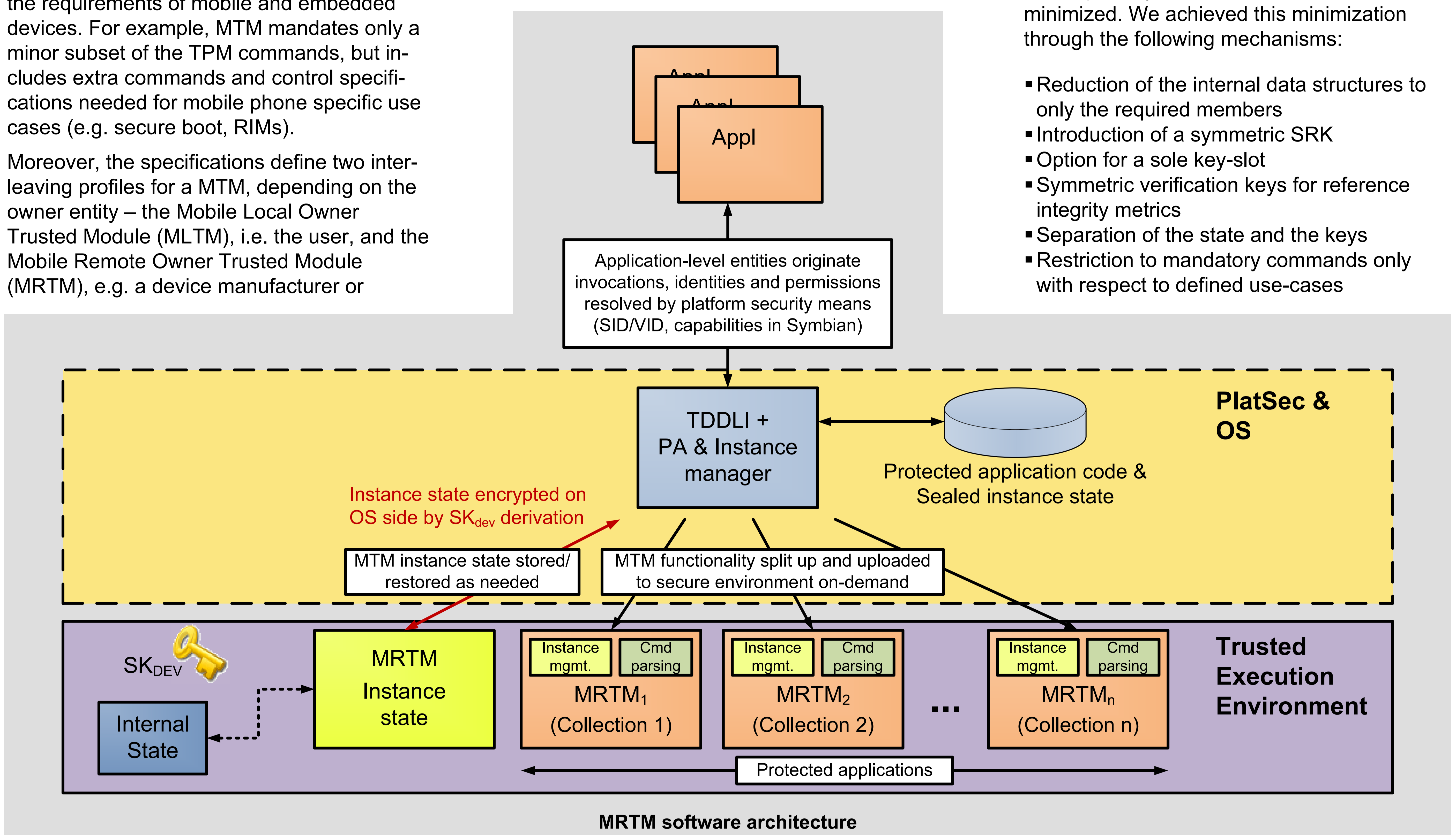
The MTM specification abstracts the device boot-up and its relation to the RTs, which are used to protect the initial boot sequence up to the place where the MTM is deployed. From this point onwards, every component measures and verifies the next component to be run against a reference integrity metric. In case a verification fails, the boot procedure is aborted, so it is not possible to boot into an untrusted system state (*secure boot*).

In our target device with TI M-Shield, the native handset boot starts executing from an on-chip ROM, which in turns loads a signed boot-loader and executes it on successful verification. The M-Shield architecture additionally includes on-chip RAM to be used by so-called *protected applications* (PAs). These can either be persistently present on an on-chip ROM or

## Architecture/Implementation

The figure shows the current design for our MRTM. Due to size constraints of the secure environment in the target devices (ARM 9 processor platforms with TI M-Shield), both the software and the state of the MRTM are “disembodied” as presented by Kursawe and Schellekens in recent papers. The MRTM commands have been grouped by size and function into collections in the form of PAs. Depending on the command to be executed, one of these collections is loaded into the secure environment and executed. In a similar fashion, the state for the MRTM(s) is loaded and returned with every command invocation. An operating system component, the Instance Manager, handles the selective loading of command collections and the state into the secure environment. Because a single PA can only carry 7kB of load, the command sizes and especially the state size have to be minimized. We achieved this minimization through the following mechanisms:

- Reduction of the internal data structures to only the required members
- Introduction of a symmetric SRK
- Option for a sole key-slot
- Symmetric verification keys for reference integrity metrics
- Separation of the state and the keys
- Restriction to mandatory commands only with respect to defined use-cases



network service provider. Support for several local and remote owners is provided by the specifications through the opportunity of deploying parallel MTMs on one device. To enable parallelism and to make MTM deployable on contemporary hardware, the specifications allow the functionality to be implemented as software by introducing new *Roots of Trust* (RTs) that mandate security properties in terms of isolation and integrity. These requirements can be met by processor security architectures like ARM TrustZone or TI M-Shield.

be uploaded to on-chip RAM as signed binaries. The system implements a firewall/monitor entry point for executing these applications, and this firewall takes care of disabling or clearing all security-critical processor features for the duration of the trusted execution environment invocation. In this manner the system provides hardware-enforced isolation for the PAs. Furthermore, the PAs have access to a limited amount of persistent secret data and to cryptographic accelerator primitives. These primitives can be used to fulfil the MTM RTs and the high-level security requirements of the specifications.

## Further Work

We still foresee further optimizations opportunities. Analysis of the state data and handling indicates room for gains both in terms of memory consumption and execution speed. Improved or better utilized hardware support within the trusted execution environment can further minimize the MRTM footprint. For devices, where strict interoperability is not an issue, trade-offs in favor of a smaller size can be achieved, e.g. through the byte-ordering of the data structures.