# Chapter 1

# Implementation

## 1.1 Attestation TA

### Trusted application

**Hashing** the memory pages of the text section of a program is the very first step the TA does when attesting a program.

**Storing reference values** needs to be done when the device is in a known secure state, the memory pages will be hashed and these hashes are stored for later comparison.

**Comparing** the hash of a memory page with its reference value is the actual attesting step, from this comparison it should be clear whether the integrity of the memory page has been violated.

**Notifying** the user is the final step to inform them about the problem to allow them to take action, this could be rebooting to ensure a secure known state again or ask for help from a specialist.

### NW OS dependencies

**Retrieving address** needs to be done from within the NW OS at the moment, this is because the datastructures that contain this information are owned by the NW OS.

**Translating address** is another functionality for which the rich OS is used, this is also due to the fact that these translations are easily determined from the NW OS datastructures.

## Extensions

**Trusted IO** could be used to inform the user of the problem (which program has been tampered with for instance), it could also take on a more coarse grained form that an led licht signals the user that some piece of software has failed the attestation which is easier but less usefull.

**Becoming independent** from the rich OS in the normal world seems like a very important step because otherwise OS/Firmware attacks are still a threat.

**Detailed attestation** is necessary, lots of software attacks are based on the used datastructures and don't impact the text section of code of programs.