4.3 Program Properties

One of the results generated from our P9 project [39] was a security analysis of OpenTitan. The security analysis contained security policies, security goals, and security mechanisms for the OpenTitan initial boot code (`mask_ROM` and `ROM_EXT`). The security policies and security goals relevant to the `mask_ROM` stage are cited below (directly taken from [39]). We will in this section present program properties, that are more refined and specific to the implementation level code of the `mask_ROM` stage and thus easier to translate into CBMC assertions. These program properties are derived from their respective security goal/policy. PROPERTY 0 is the exception to this as it has no direct parent goal/policy, but is still fundamental for overall program correctness and safety.

"**No parent policy or goal**"

- PROPERTY 0: The `mask_ROM` boot code must be free of bugs.

"**P1: It should only be possible to execute code that has been validated (authenticity/integrity)**"

"**G1: The hash of the `ROM_EXT` image and the signature of the hash must be validated by `mask_ROM` before it is executed to ensure authenticity and integrity of the image.**"

- PROPERTY 1: The `ROM_EXT` manifest for a `ROM_EXT` must be signed with a RSA-3072 signature. If a `ROM_EXT` manifest for a `ROM_EXT` is unsigned (i.e. the signature is a sequence of zeros) the `ROM_EXT` is considered invalid to boot from.

- PROPERTY 2: The public RSA-3072 key used for the signature contained in the `ROM_EXT` manifest, must be valid in order to be considered valid to boot from.

- PROPERTY 3: The HMAC hash must be calculated by either a SHA2-256, SHA3-256, SHA3-384, or SHA3-512 hash function.

- PROPERTY 4: The computed HMAC hash message must be calculated from `system_state_value || device_usage_value || signed_area(rom_ext)` [36].

- PROPERTY 5: The signature in `ROM_EXT` manifest must be validated using the `RSASSA-PKCS1-V1_5-VERIFY` [45] function with inputs: public RSA-3072 key, appended message (`system_state_value || device_usage_value || signed_area(rom_ext)`), and RSA-3072 signature. If the function returns false the `ROM_EXT` is invalid to boot from.

- PROPERTY 6: If all validation steps have succeeded then transfer execution to `ROM_EXT` by starting execution at the entry point of `ROM_EXT` image code. If execution returns, execute the `fail_rom_ext_returned` function provided by the boot policy.

- PROPERTY 7: If at any point a `ROM_EXT` is invalidated the `ROM_EXT` is considered unsafe to boot from and the `mask_ROM` must proceed to validate the next `ROM_EXT` .

- PROPERTY 8: If validation fails for all the `ROM_EXTs`, `mask_ROM` must execute the `fail` function provided by the boot policy.

"**P4: There is a privilege hierarchy that is respected (i.e. access rights: read, write, and execute)**"

"**G10, G11, G12: Only software with write/read/execute access to some memory section may modify/read/execute it.**"

- PROPERTY 9: The entire flash must be covered by a PMP region at the initialization of `mask_ROM`. The PMP region must be locked and restricted to read-only access.

- PROPERTY 10: If a `ROM_EXT` is validated then `mask_ROM` must create a PMP region covering the `ROM_EXT` memory, that is locked and that allows for read and execution access.