RIOT OTA

Over (Thin) Air Update

OTA Problem Domains

1. ‘a way to manage software updates over network’
2. ‘a way to replace the running code with new code’

this presentation focuses on domain 2, specifically common infrastructure, and core features for RIOT.

Identified use cases

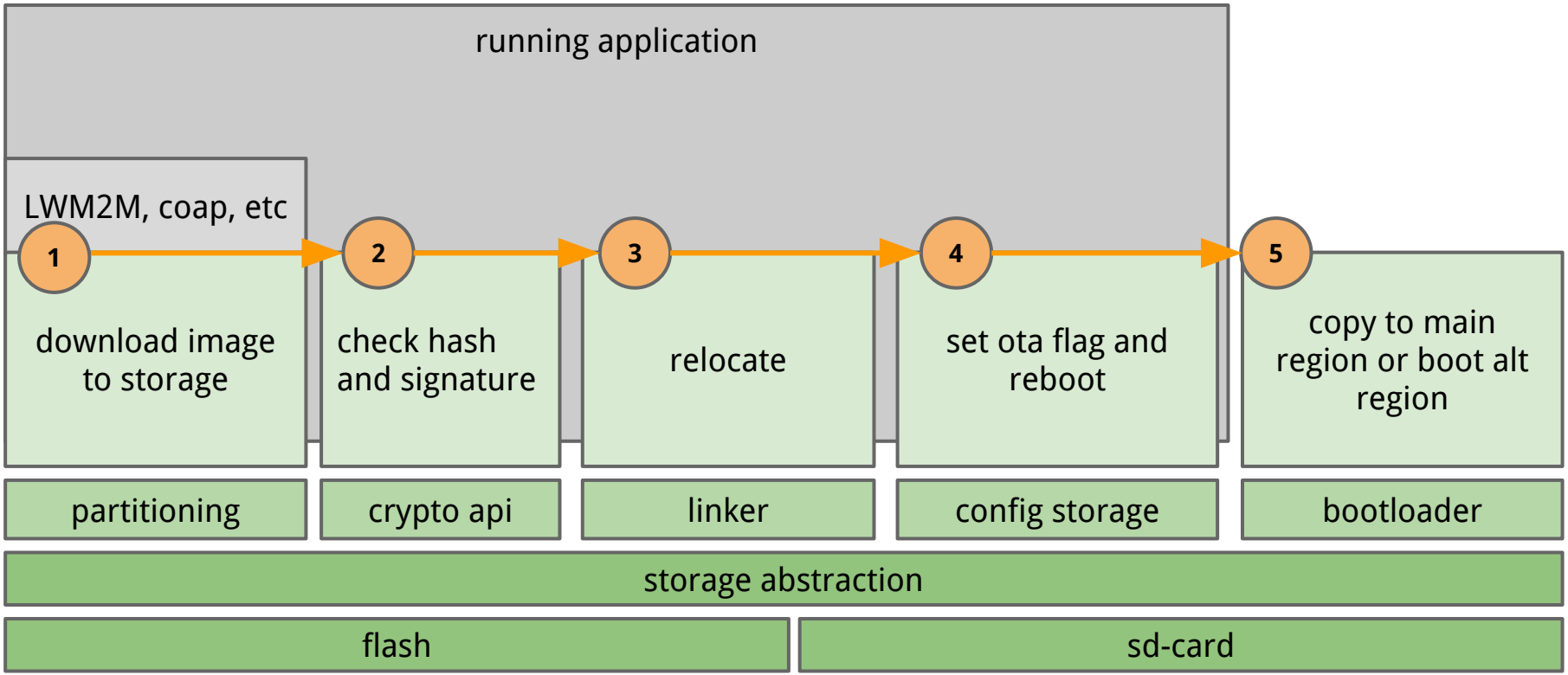
- separate update code from main code
- download with any high level protocol
- encrypted and signed images
- protection against power loss during flash
- sometimes limited internal flash

Proposed solutions

- with enough internal flash
  - use two regions, have a bootloader switch
  - if new image won’t boot, go back to old one

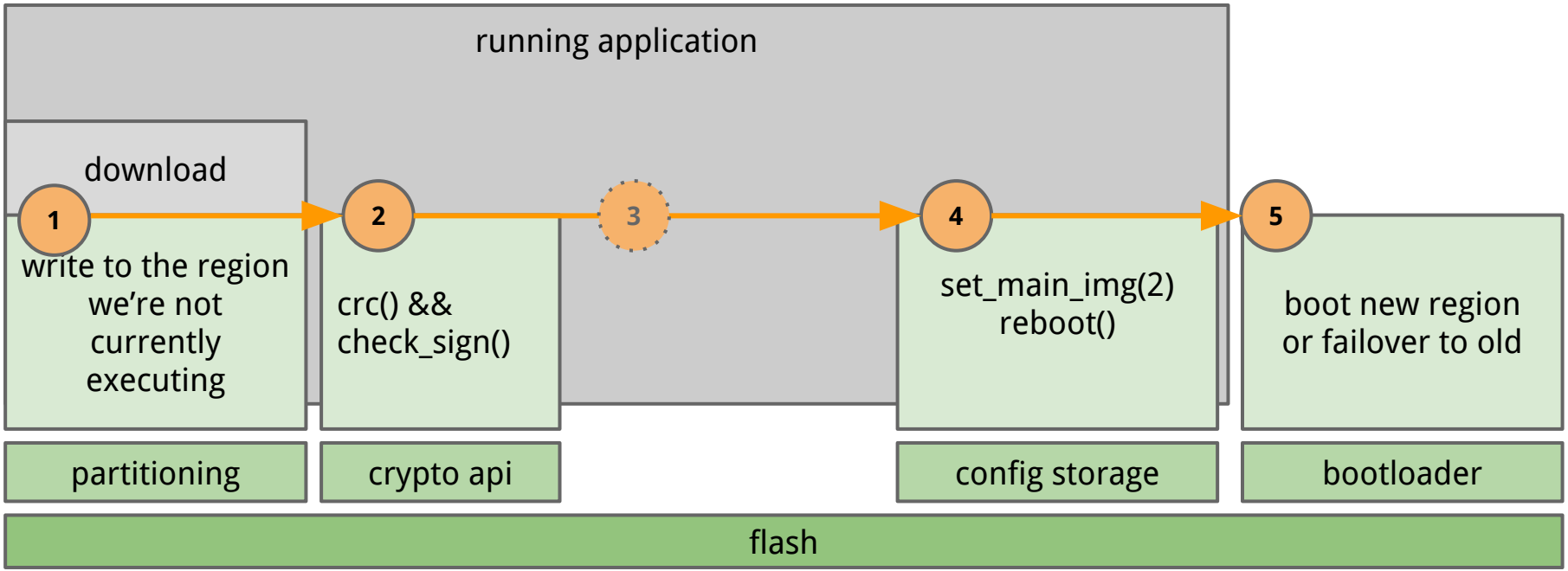
- with external flash
  - download from running app to external flash
  - bootloader copies from external to internal

Architecture Proposal



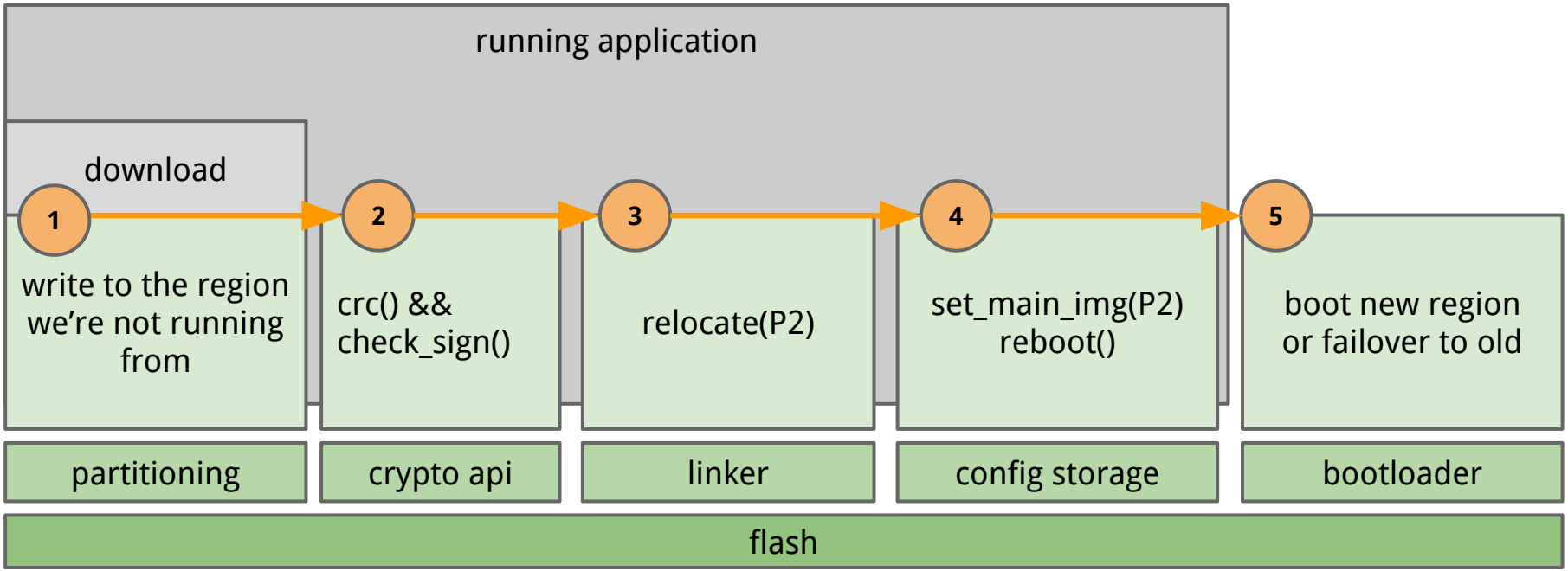
double internal flash

protection against non-bootable OTA by keeping the old image bootable



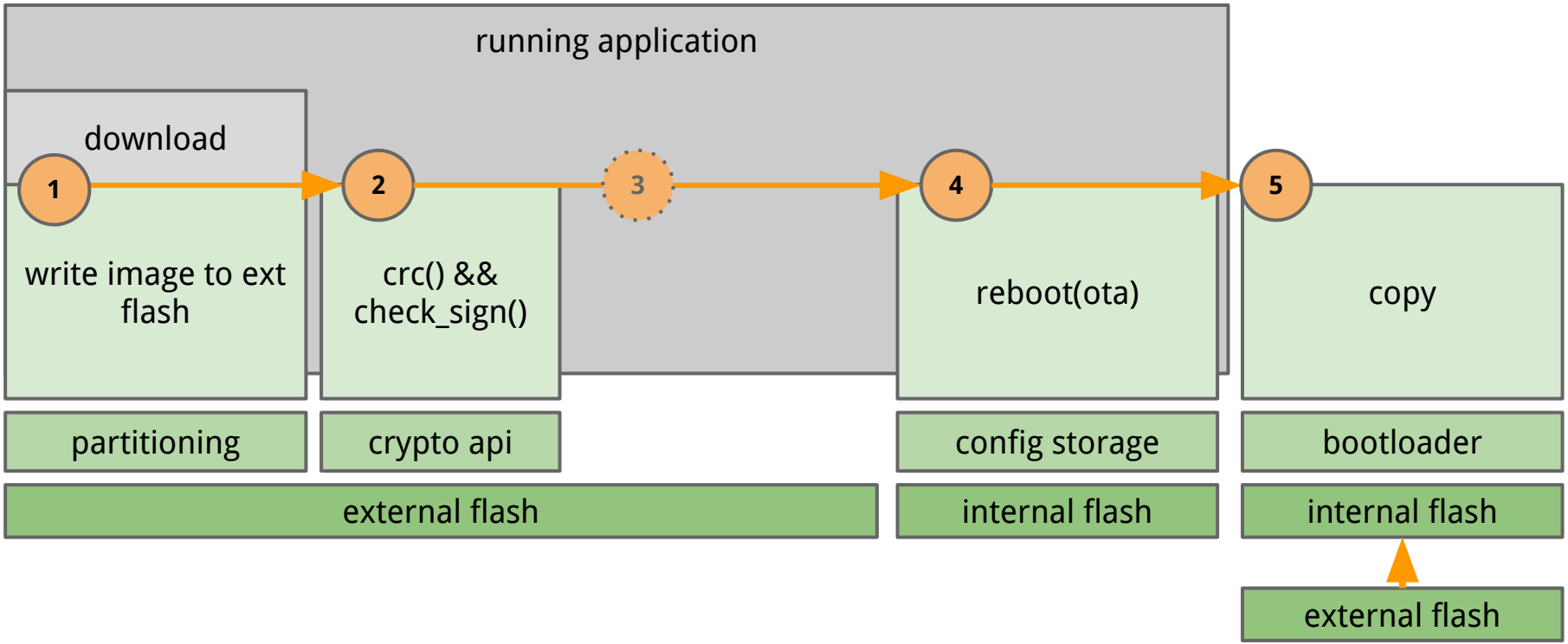
double internal flash + relocation

relocate on-device to avoid having multiple images



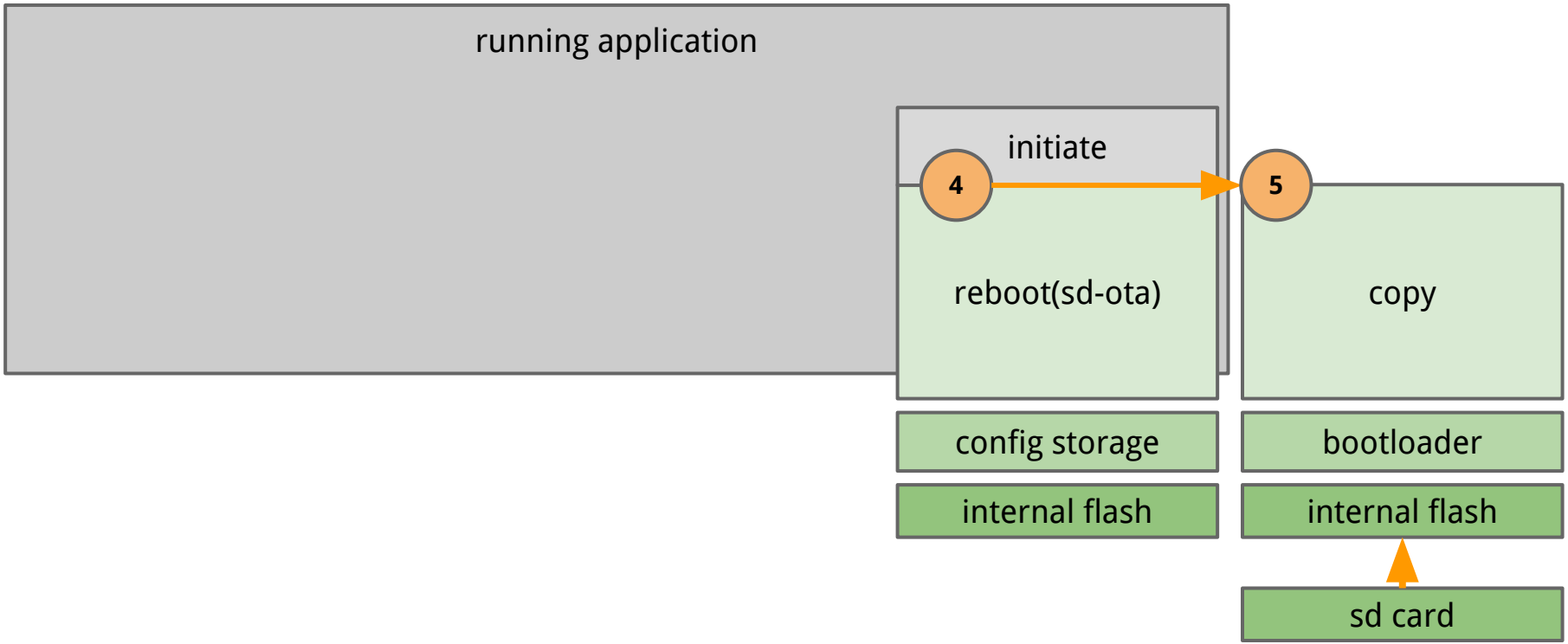
external flash, no relocation

not enough internal flash for secondary image, but other non-bootable storage available



from sd card

offline update from sd-card



RIOT components

1. partitioning / named regions in board config
2. internal and external flash api
3. cryptography, hash and signature

- binary relocation

4. image format with relocation info
5. permanent key/value store for bootreason

- bootloader with board/app specific drivers like SPI

Relevant PRs so far

crypto

chacha cipher https://github.com/RIOT-OS/RIOT/pull/2387

block ciphers (for DTLS) https://github.com/RIOT-OS/RIOT/pull/1342

flash

nvram api https://github.com/RIOT-OS/RIOT/pull/2353

flash api https://github.com/RIOT-OS/RIOT/pull/2239

relocation

elfloader https://github.com/RIOT-OS/RIOT/pull/1421