Genode's TrustZone demo on the USB Armory
story, design, and use

Martin Stein
<martin.stein@genode-labs.com>

Outline

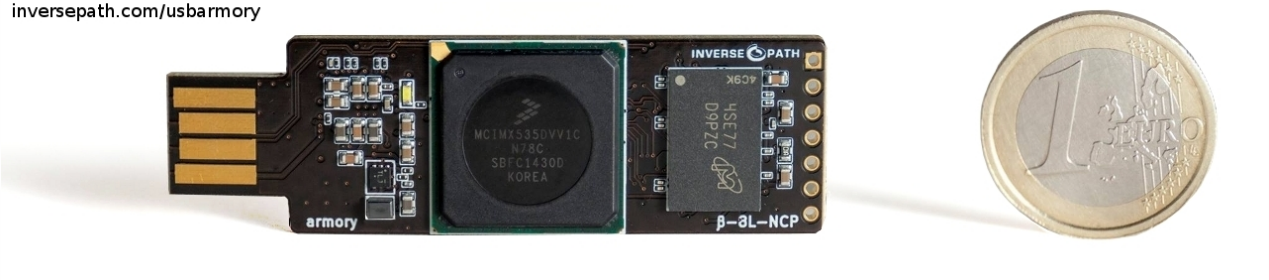
1. Background and Motivation
2. Implementation
3. Results

Outline

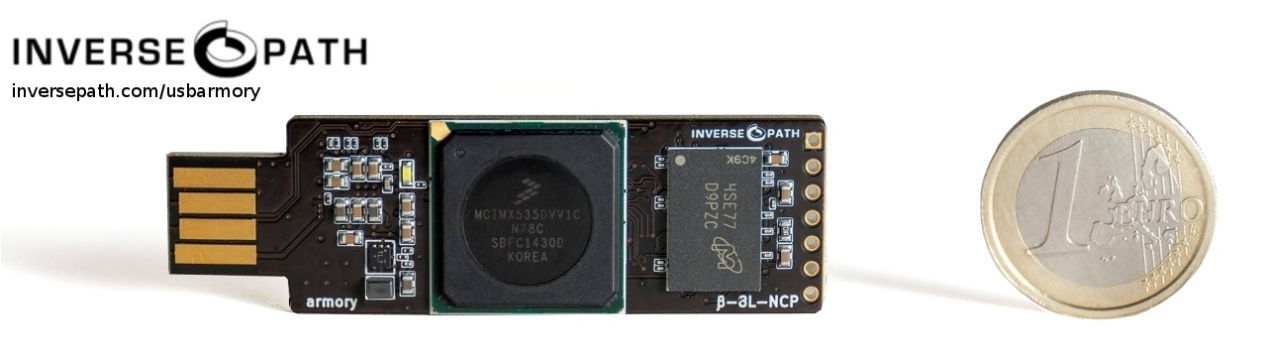
What is the USB Armory?

INVERSE PATH

inversepath.com/usbarmory

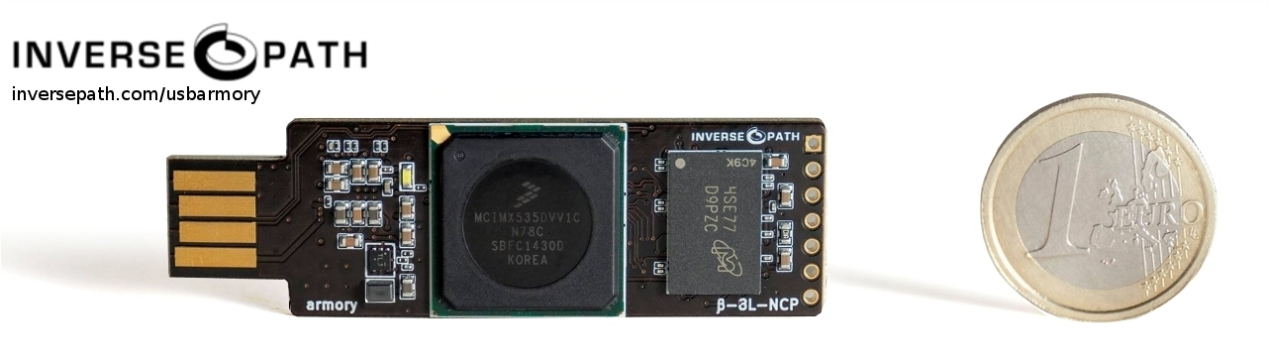


What is the USB Armory?



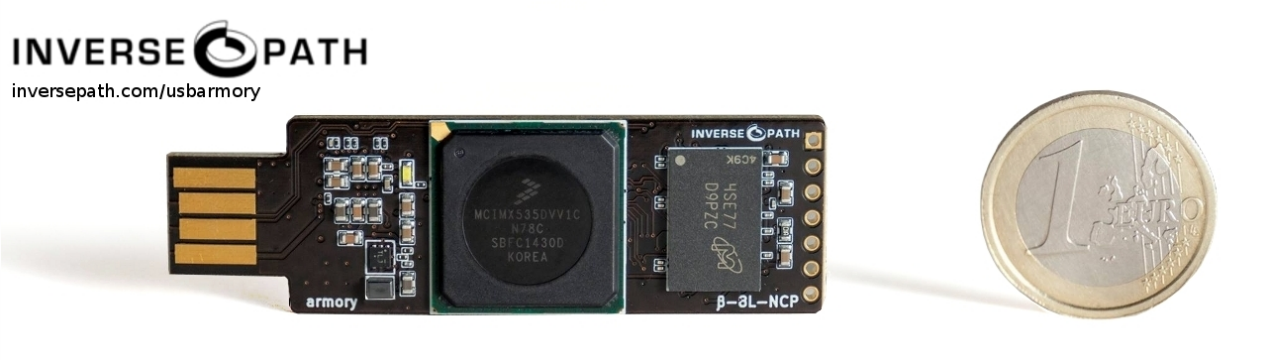
- USB stick computer that runs standard Linux distributions

What is the USB Armory?



- USB stick computer that runs standard Linux distributions
- Open design featuring i.MX53, TrustZone, Secure Boot, and an LED

What is the USB Armory?



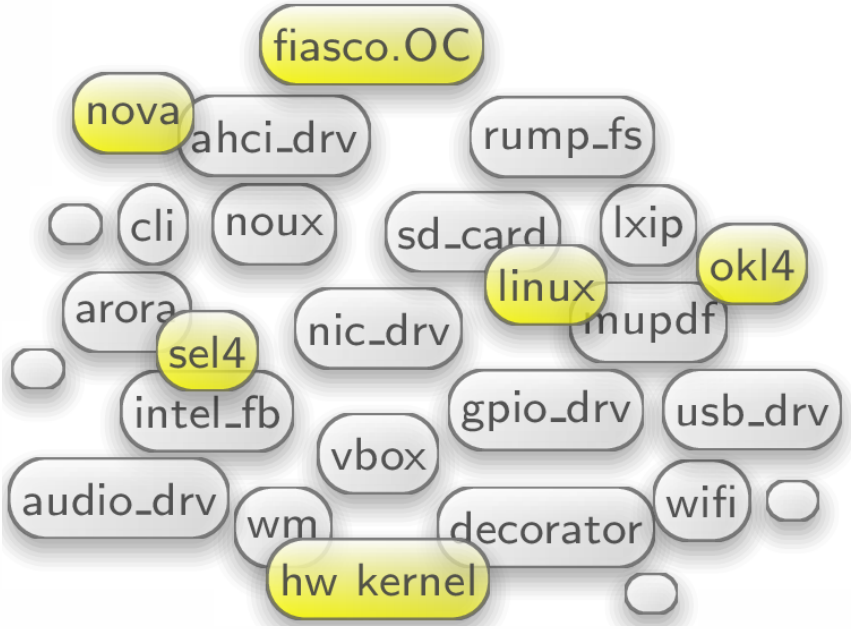
- USB stick computer that runs standard Linux distributions
- Open design featuring i.MX53, TrustZone, Secure Boot, and an LED
- Communicate with Linux via Ethernet

What is Genode?

Microkernels
Capabilities
Componentization
Virtualization

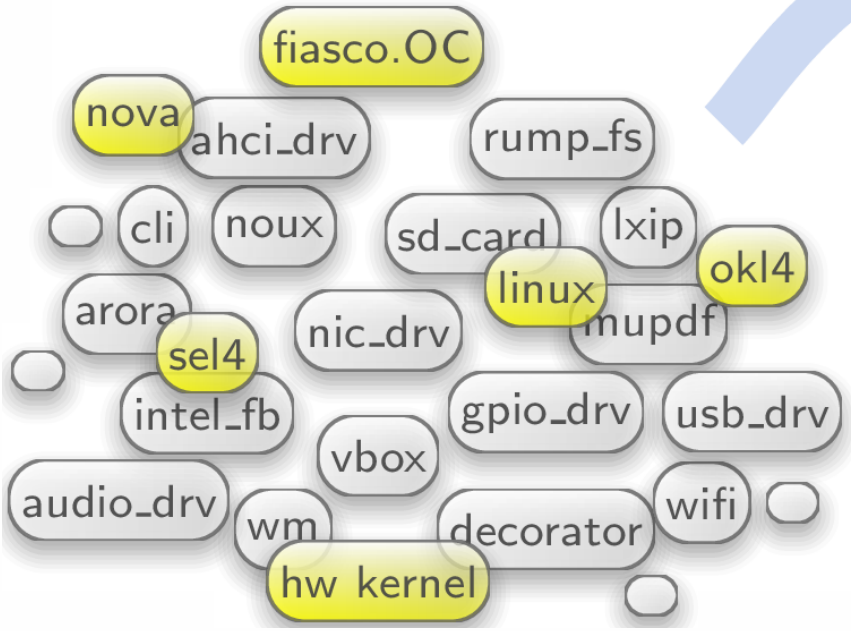
What is Genode?

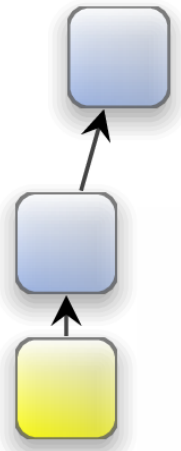
Microkernels
Capabilities
Componentization
Virtualization



What is Genode?

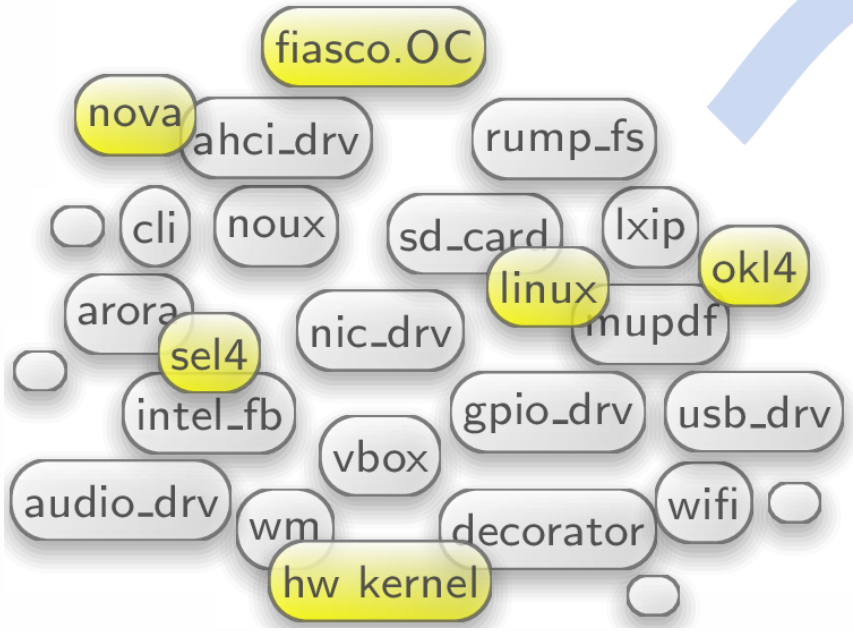
Microkernels
Capabilities
Componentization
Virtualization

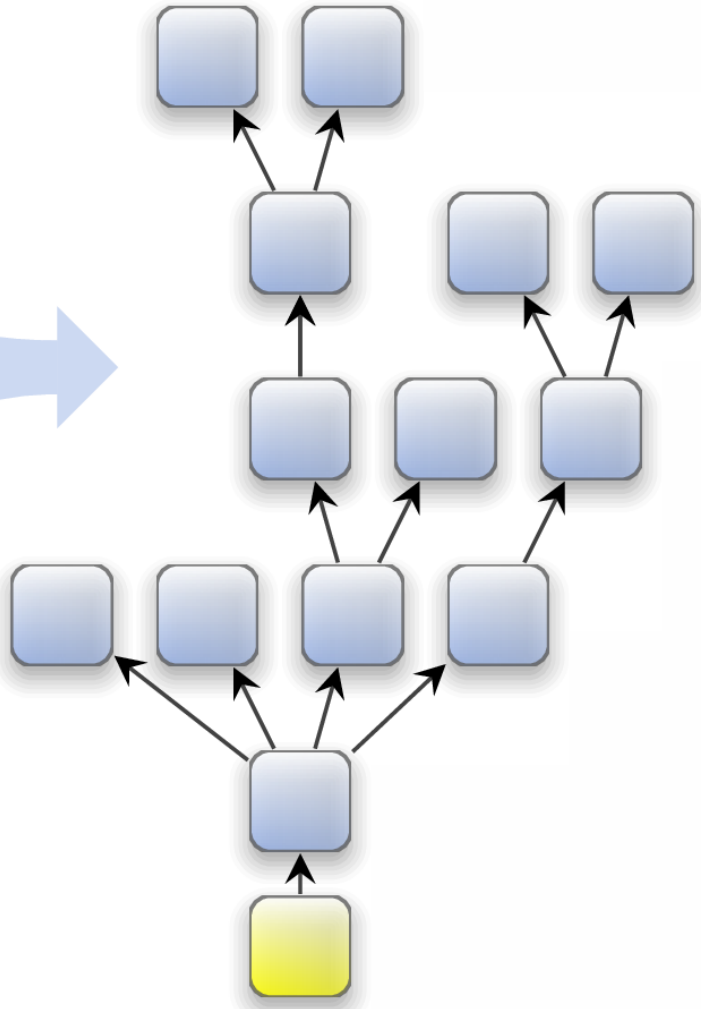




What is Genode?

Microkernels
Capabilities
Componentization
Virtualization





What is ARM TrustZone?

Separate hardware into **Secure World** and **Normal World**

What is ARM TrustZone?

Separate hardware into **Secure World** and **Normal World**

- One bit in CPU

What is ARM TrustZone?

Separate hardware into **Secure World** and **Normal World**

- One bit in CPU
- Switch to Normal world via new Monitor mode

What is ARM TrustZone?

Separate hardware into **Secure World** and **Normal World**

- One bit in CPU
- Switch to Normal world via new Monitor mode
- Switch to Secure world via SMC instruction

What is ARM TrustZone?

Separate hardware into **Secure World** and **Normal World**

- One bit in CPU
- Switch to Normal world via new Monitor mode
- Switch to Secure world via SMC instruction
- The rest is implementer specific

TrustZone on the i.MX53

Interrupt Controller

TrustZone on the i.MX53

Interrupt Controller

- Routes and injects interrupts

TrustZone on the i.MX53

Interrupt Controller

- Routes and injects interrupts

Central Security Unit

TrustZone on the i.MX53

Interrupt Controller

- Routes and injects interrupts

Central Security Unit

- Separates RAM/MMIO

TrustZone on the i.MX53

Interrupt Controller

- Routes and injects interrupts

Central Security Unit

- Separates RAM/MMIO
- Defines permissions of DMA engines

TrustZone in Genode

In our custom kernel

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM
  - Like a Thread but with special transition code

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM
  - Like a Thread but with special transition code

In the userland

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM
  - Like a Thread but with special transition code

In the userland

- Core component provides VM service to control VM

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM
  - Like a Thread but with special transition code

In the userland

- Core component provides VM service to control VM
- VMM component

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM
  - Like a Thread but with special transition code

In the userland

- Core component provides VM service to control VM
- VMM component
  - Initializes TrustZone

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM
  - Like a Thread but with special transition code

In the userland

- Core component provides VM service to control VM
- VMM component
  - Initializes TrustZone
  - Acts as Bootloader for VM OS

TrustZone in Genode

In our custom kernel

- Genode threads run in Secure world
- Normal world is a VM
  - Like a Thread but with special transition code

In the userland

- Core component provides VM service to control VM
- VMM component
  - Initializes TrustZone
  - Acts as Bootloader for VM OS
  - Controls VM context

Motivation for the USB Armory demo

Starting Point

Motivation for the USB Armory demo

Starting Point

- Basic TrustZone support with Versatile Express board

Motivation for the USB Armory demo

Starting Point

- Basic TrustZone support with Versatile Express board
- i.MX53 TrustZone support with Android Demo on SABRE Lite board

Motivation for the USB Armory demo

Starting Point

- Basic TrustZone support with Versatile Express board
- i.MX53 TrustZone support with Android Demo on SABRE Lite board

Goals

Motivation for the USB Armory demo

Starting Point

- Basic TrustZone support with Versatile Express board
- i.MX53 TrustZone support with Android Demo on SABRE Lite board

Goals

- Put Normal world Linux under the supervision of Secure world Genode

Motivation for the USB Armory demo

Starting Point

- Basic TrustZone support with Versatile Express board
- i.MX53 TrustZone support with Android Demo on SABRE Lite board

Goals

- Put Normal world Linux under the supervision of Secure world Genode
- Keep feature parity with the original USB Armory setup

Outline

First steps

Basic Linux setup with Busy Box initrd

First steps

Basic Linux setup with Busy Box initrd

- Adapted VMM to USB Armory and mainline Linux

First steps

Basic Linux setup with Busy Box initrd

- Adapted VMM to USB Armory and mainline Linux
- Skipped Linux code that is not dangerous but obstructive

First steps

Basic Linux setup with Busy Box initrd

- Adapted VMM to USB Armory and mainline Linux
- Skipped Linux code that is not dangerous but obstructive
- Shared unprotected UART between worlds

First steps

Basic Linux setup with Busy Box initrd

- Adapted VMM to USB Armory and mainline Linux
- Skipped Linux code that is not dangerous but obstructive
- Shared unprotected UART between worlds
- Adapted Linux IC driver

First steps

Basic Linux setup with Busy Box initrd

- Adapted VMM to USB Armory and mainline Linux
- Skipped Linux code that is not dangerous but obstructive
- Shared unprotected UART between worlds
- Adapted Linux IC driver
- Adapted Linux RAM configuration

Towards a broader feature set

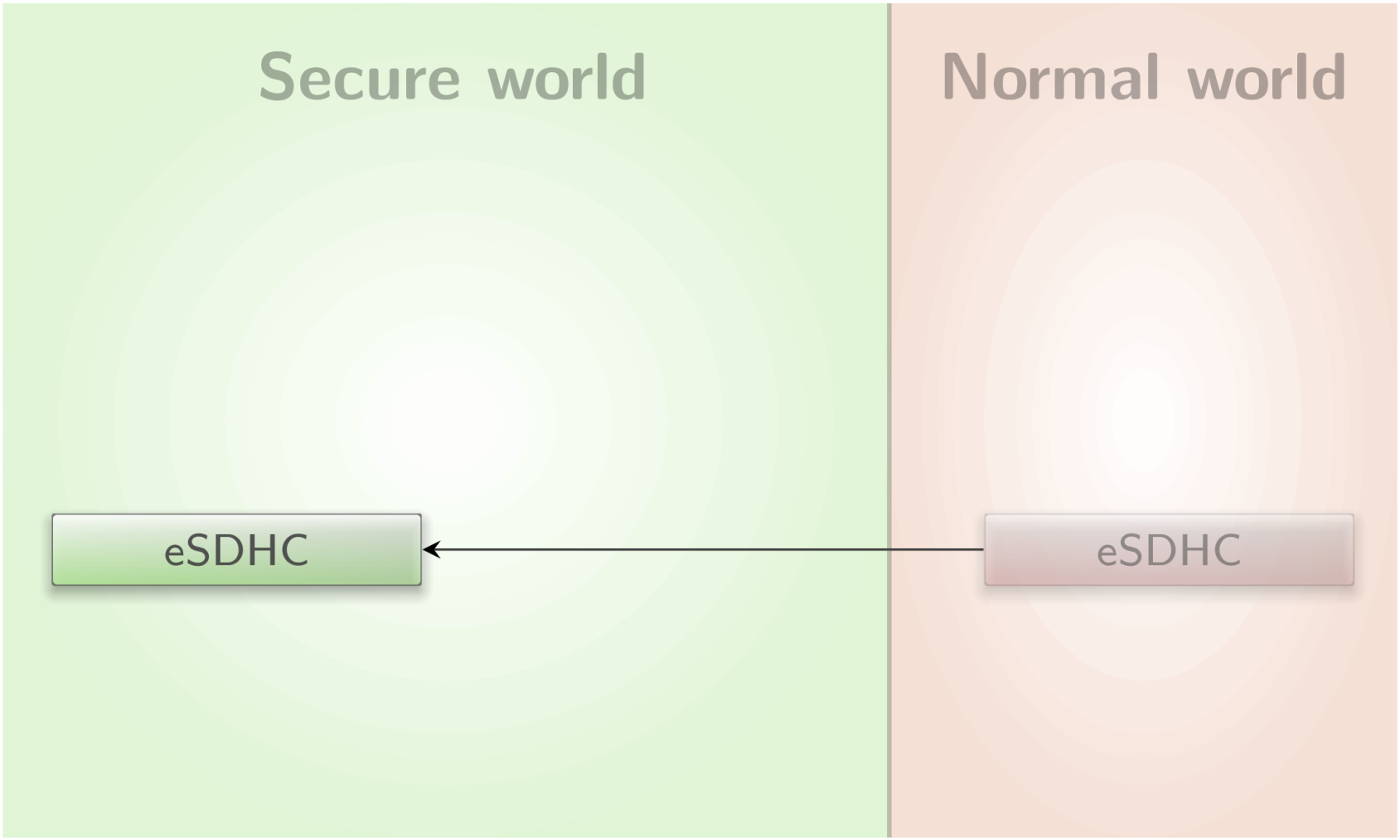
Provide a rich rootfs **and** protect the eSDHC

Towards a broader feature set

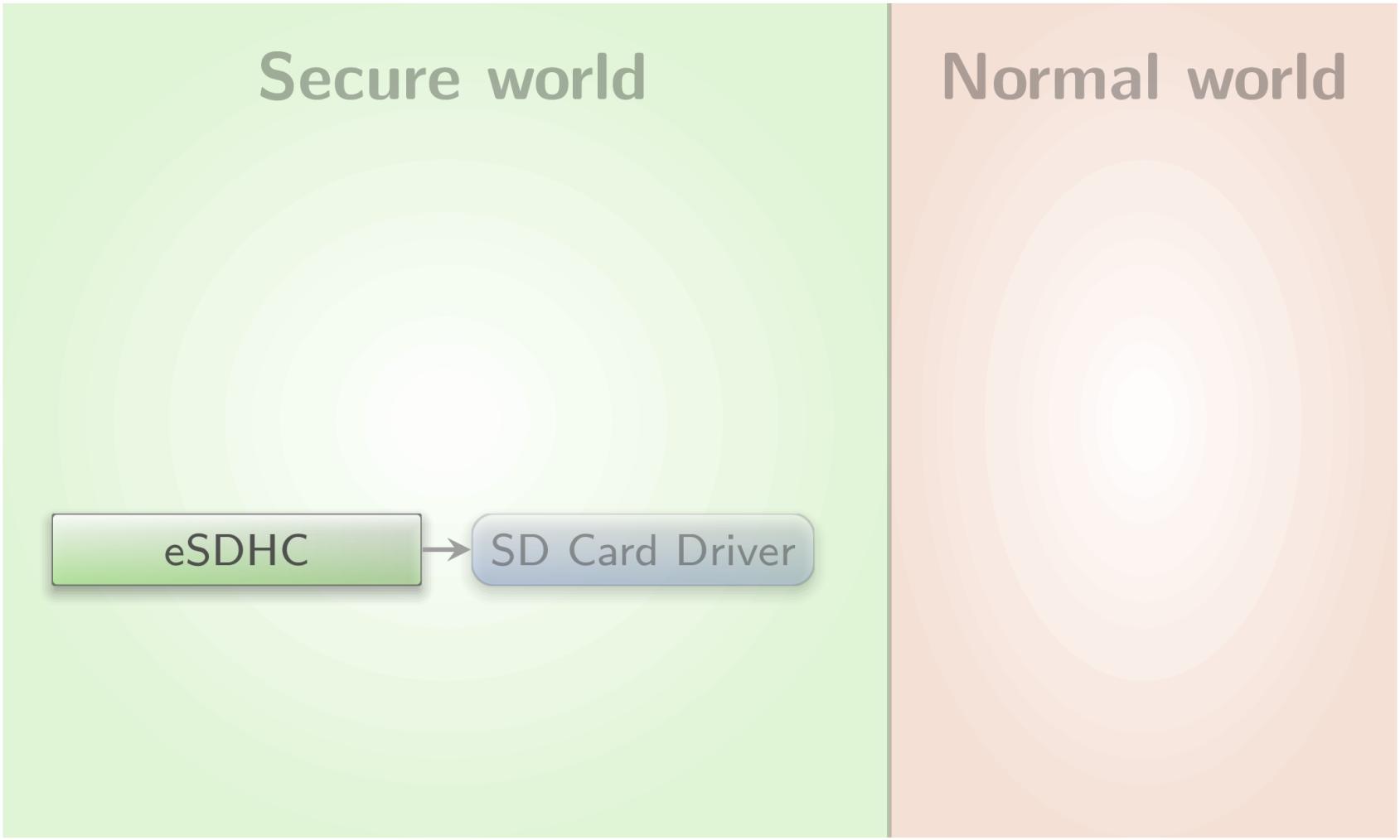
Provide a rich rootfs **and** protect the eSDHC

→ Para-virtualized block driver

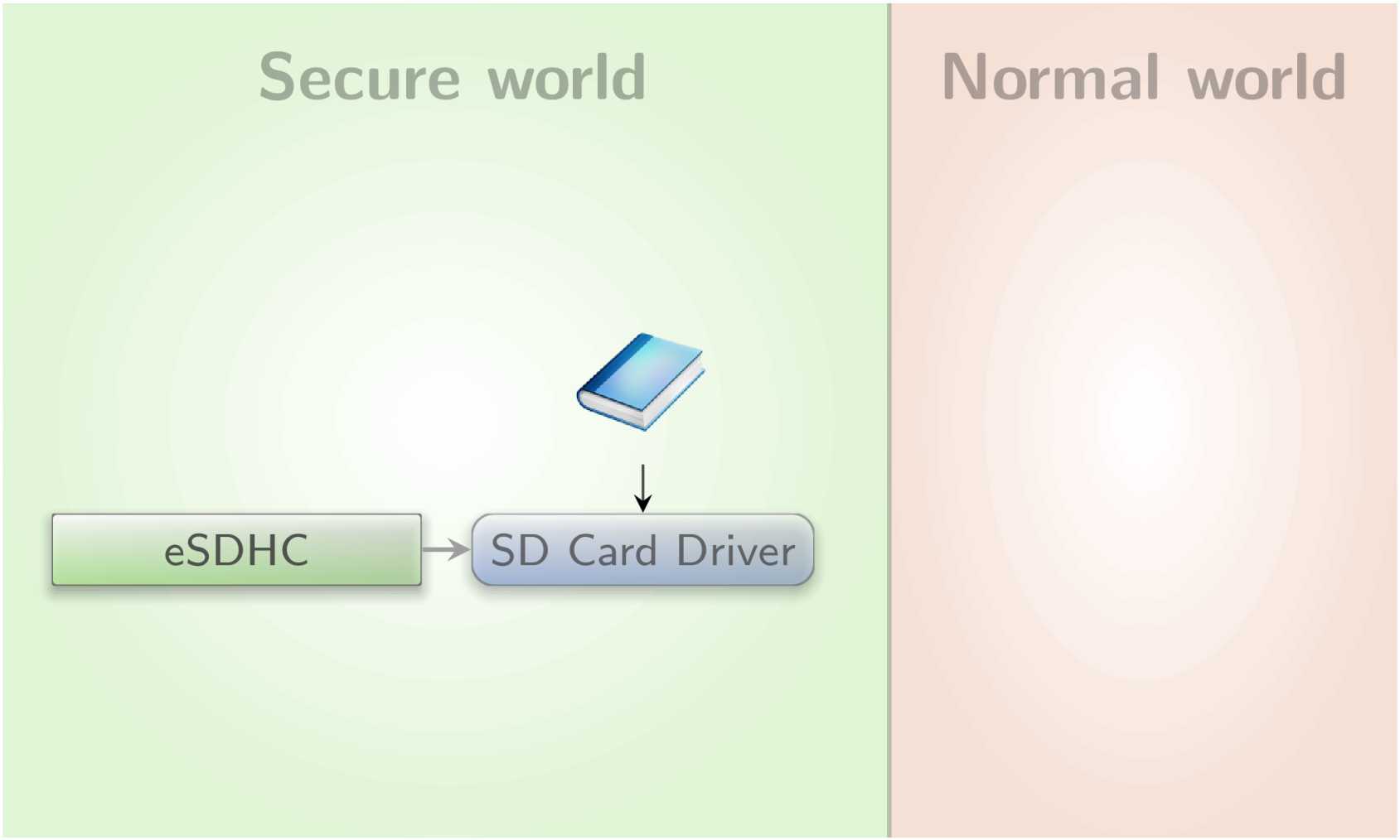
Separation of the SD card



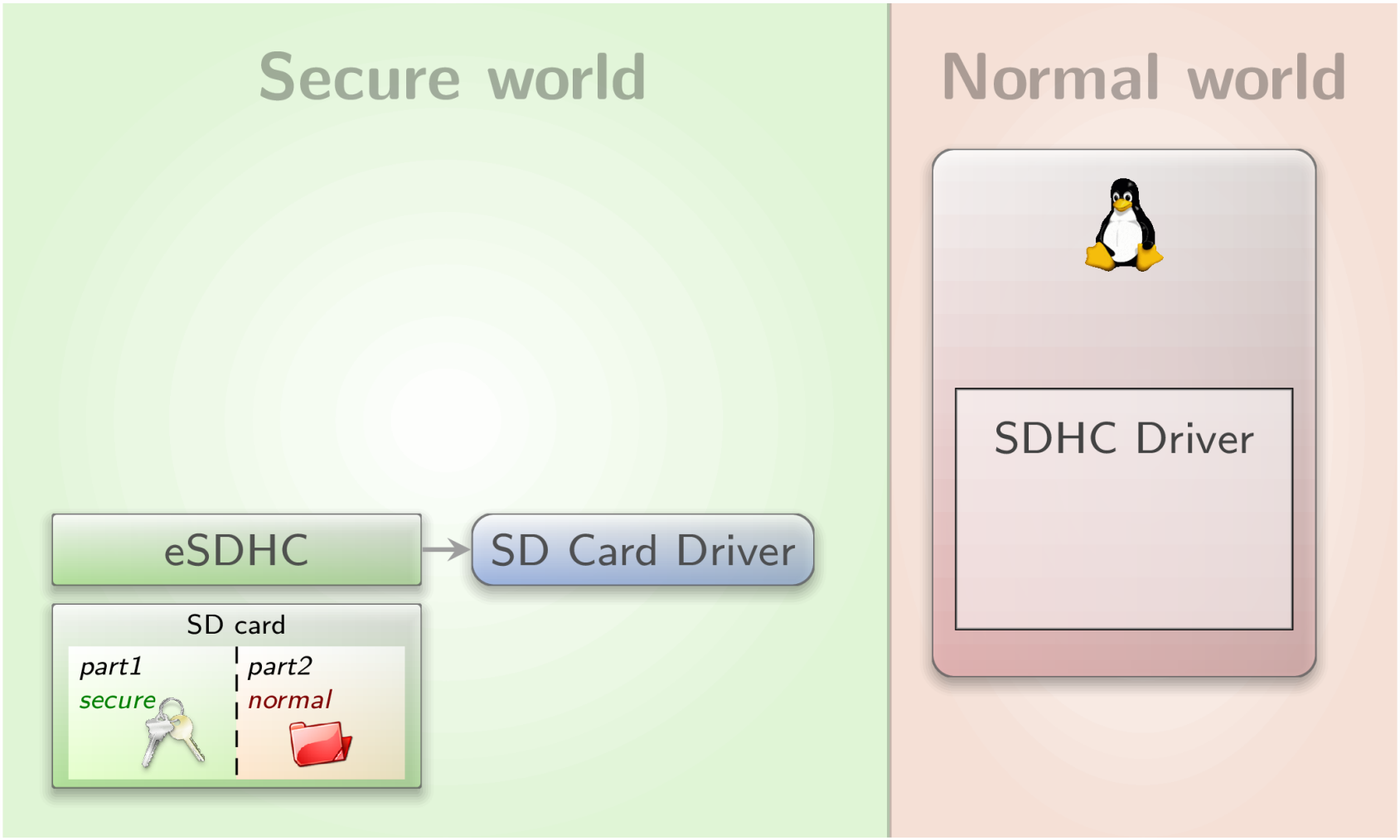
Separation of the SD card



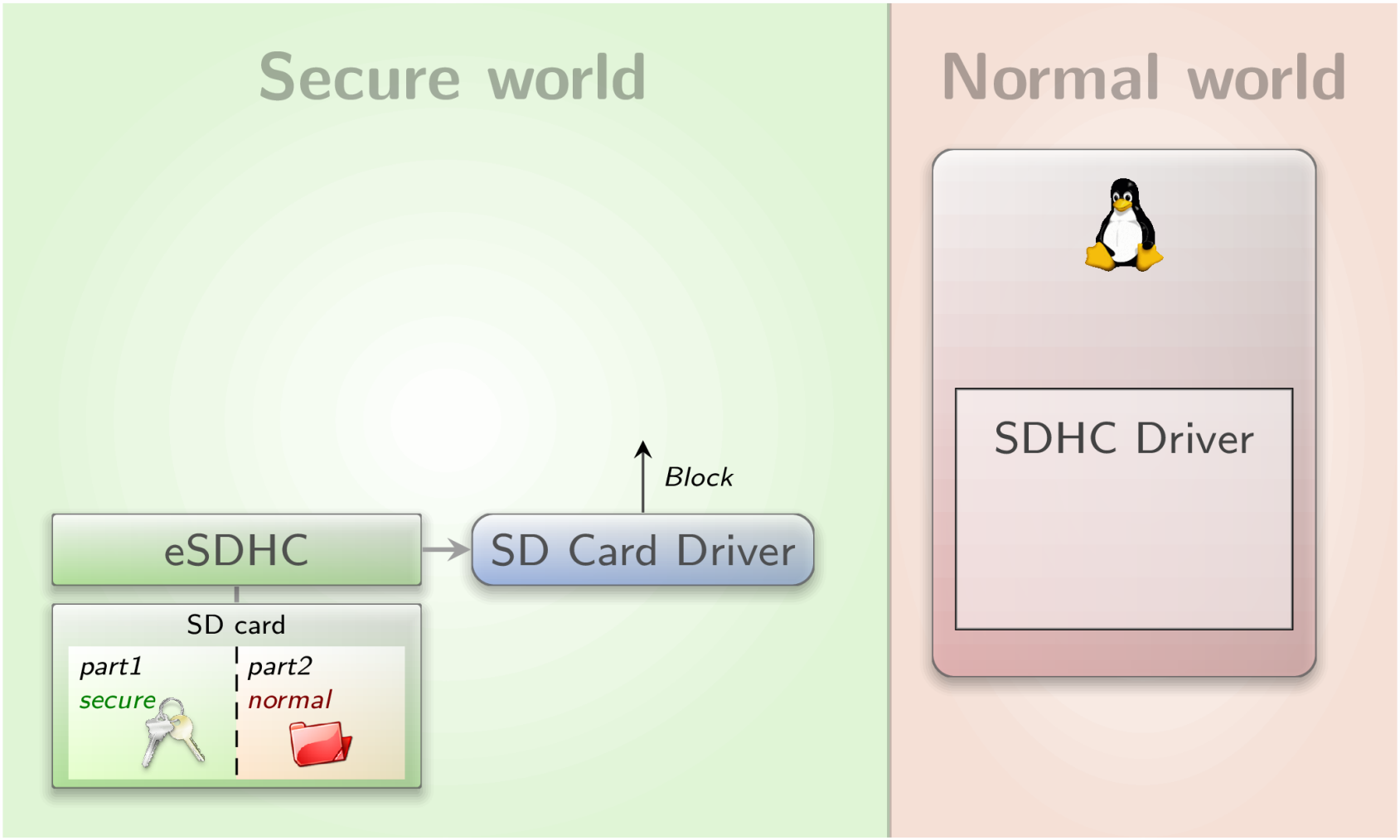
Separation of the SD card



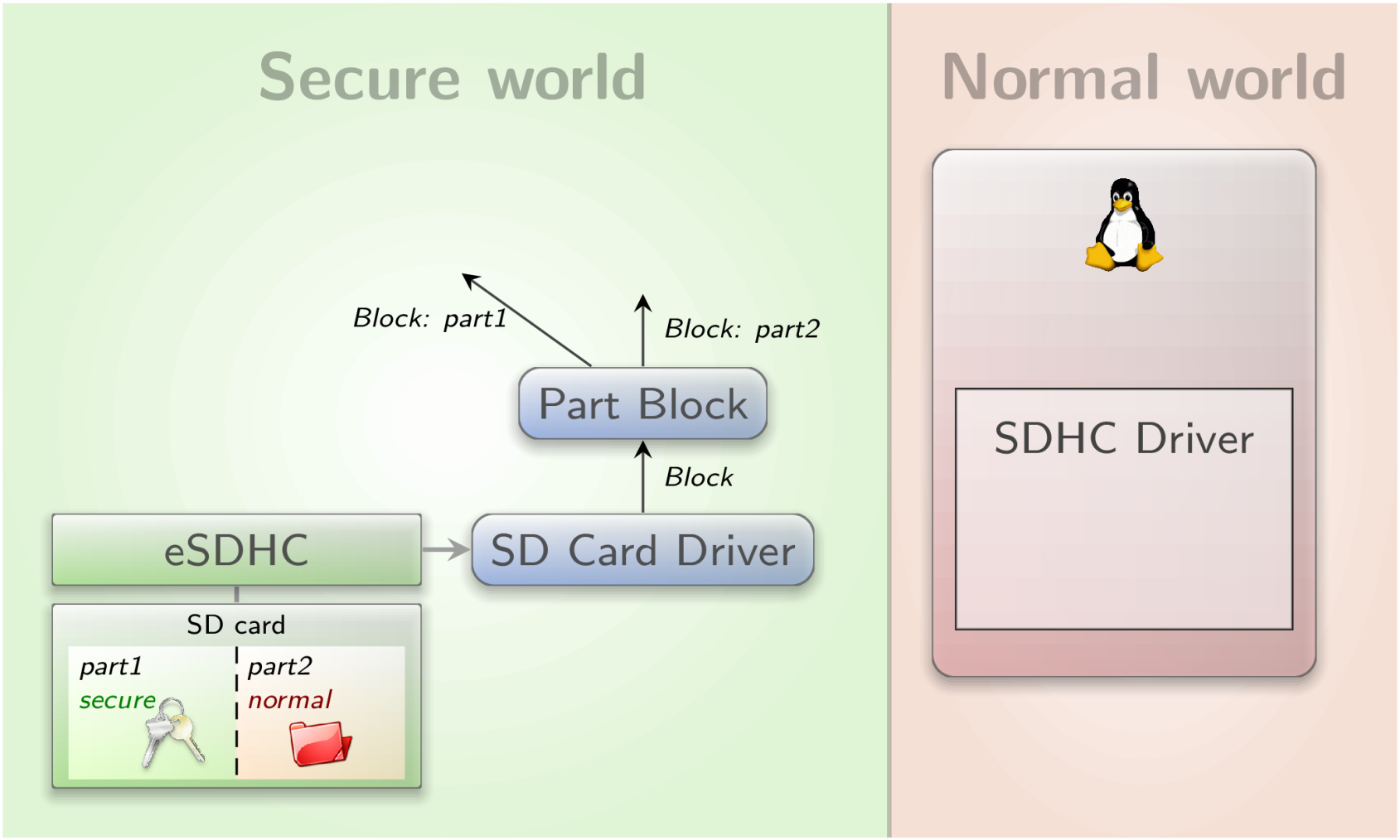
Separation of the SD card



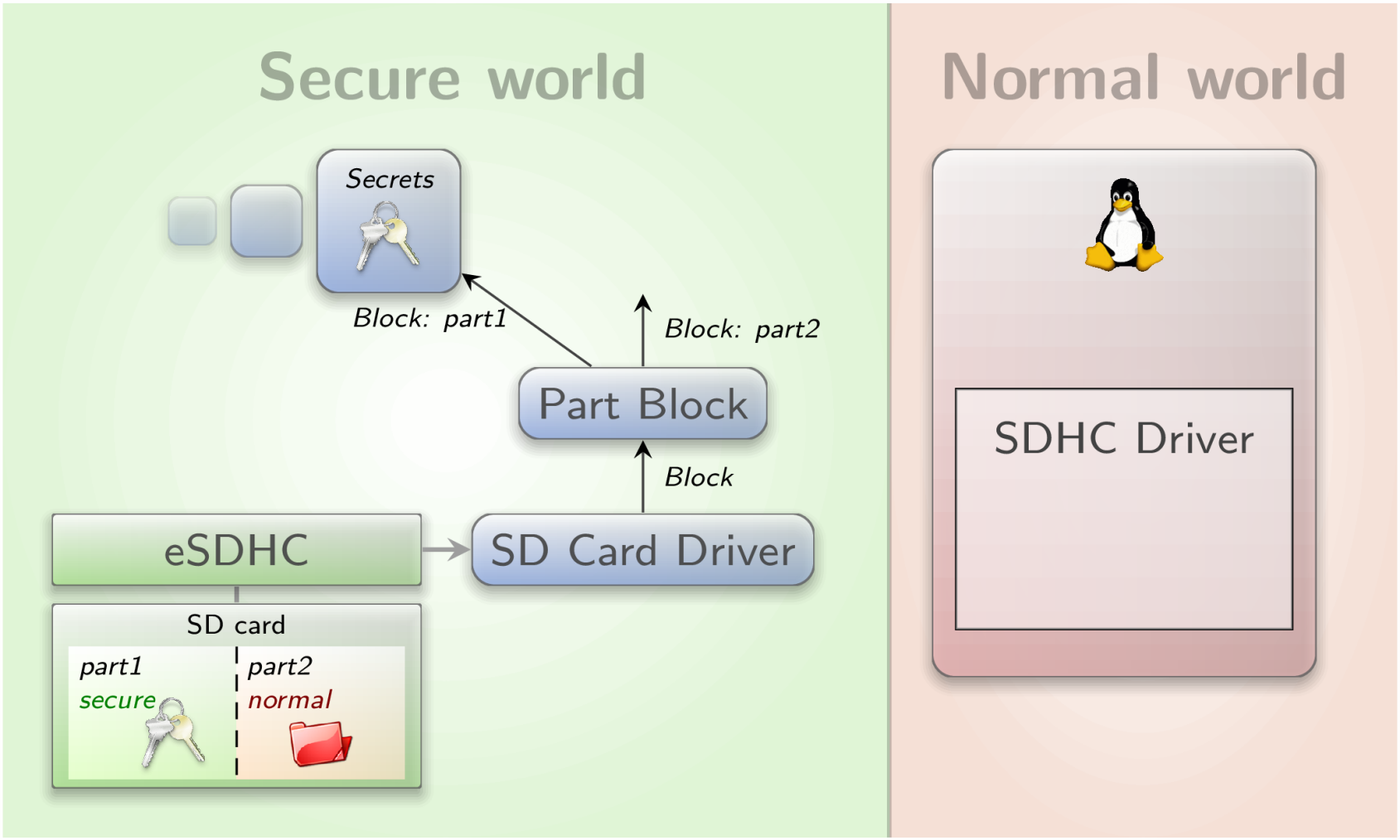
Separation of the SD card



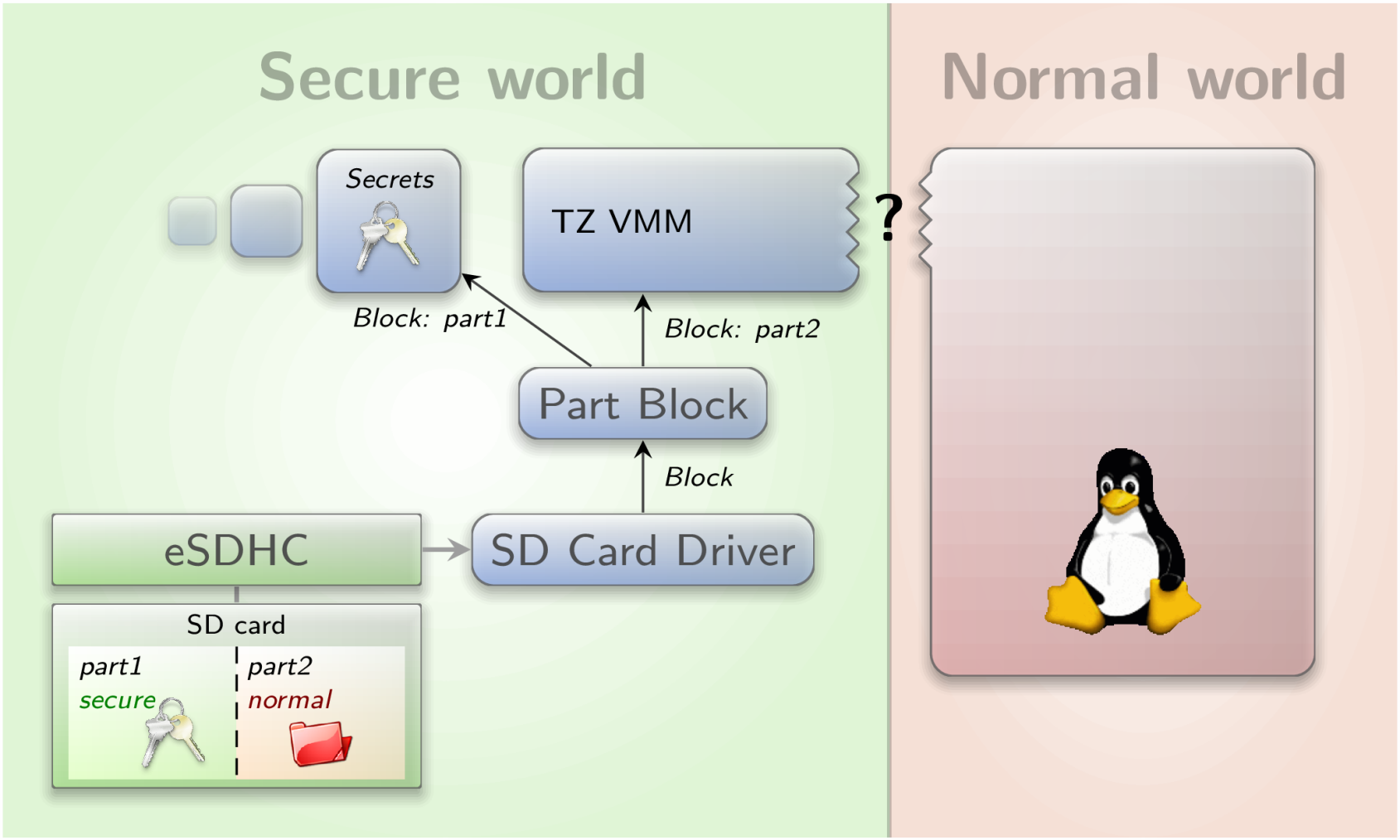
Separation of the SD card



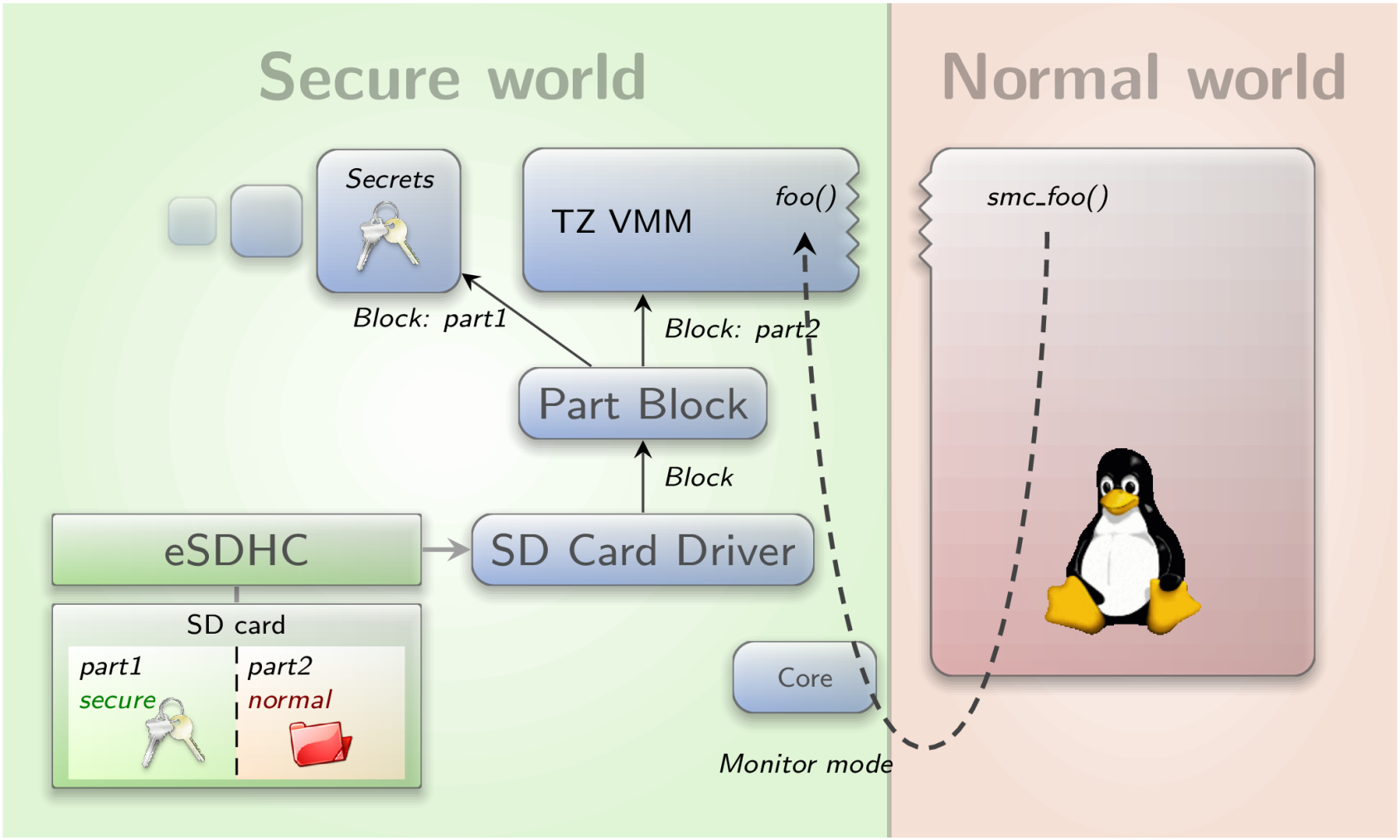
Separation of the SD card



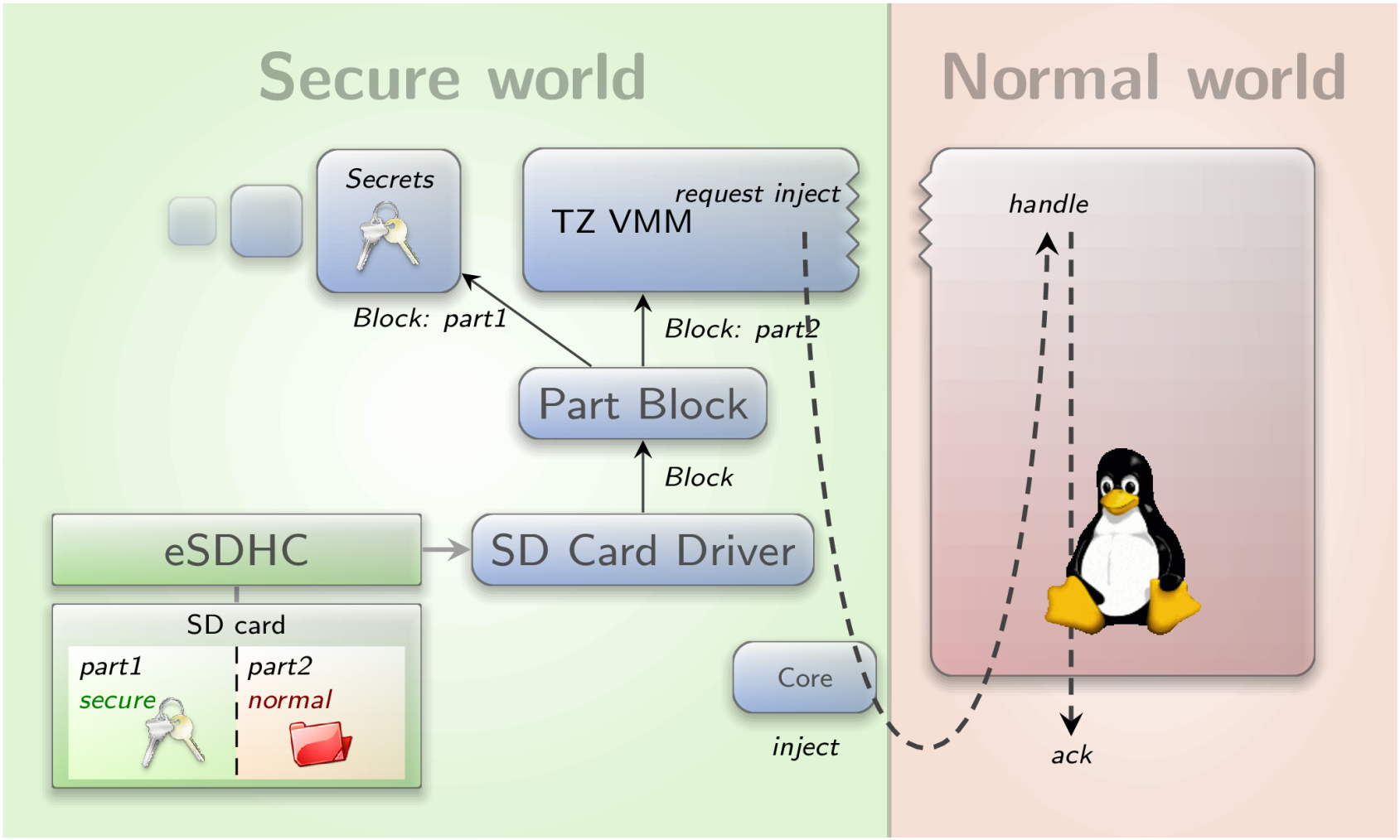
Separation of the SD card



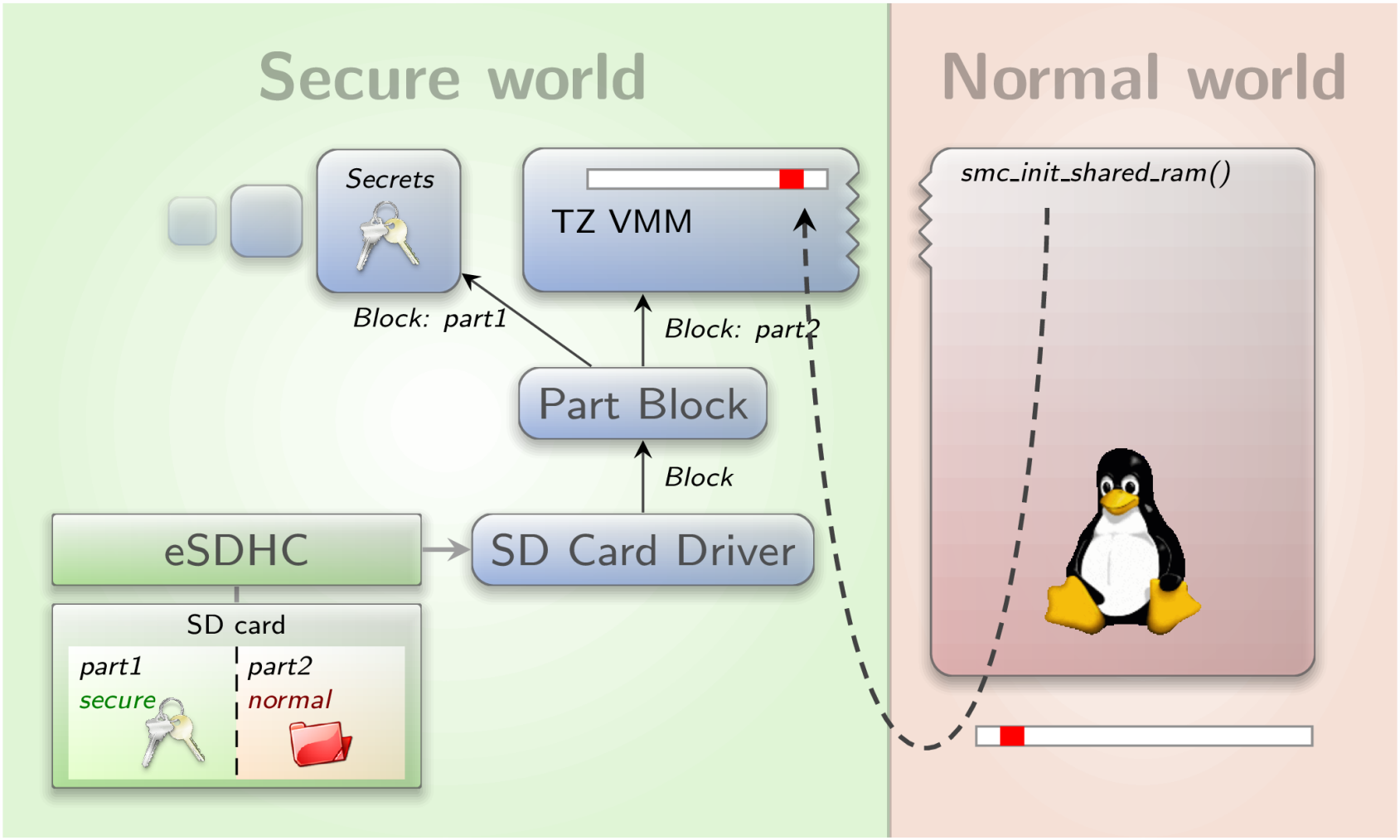
Separation of the SD card



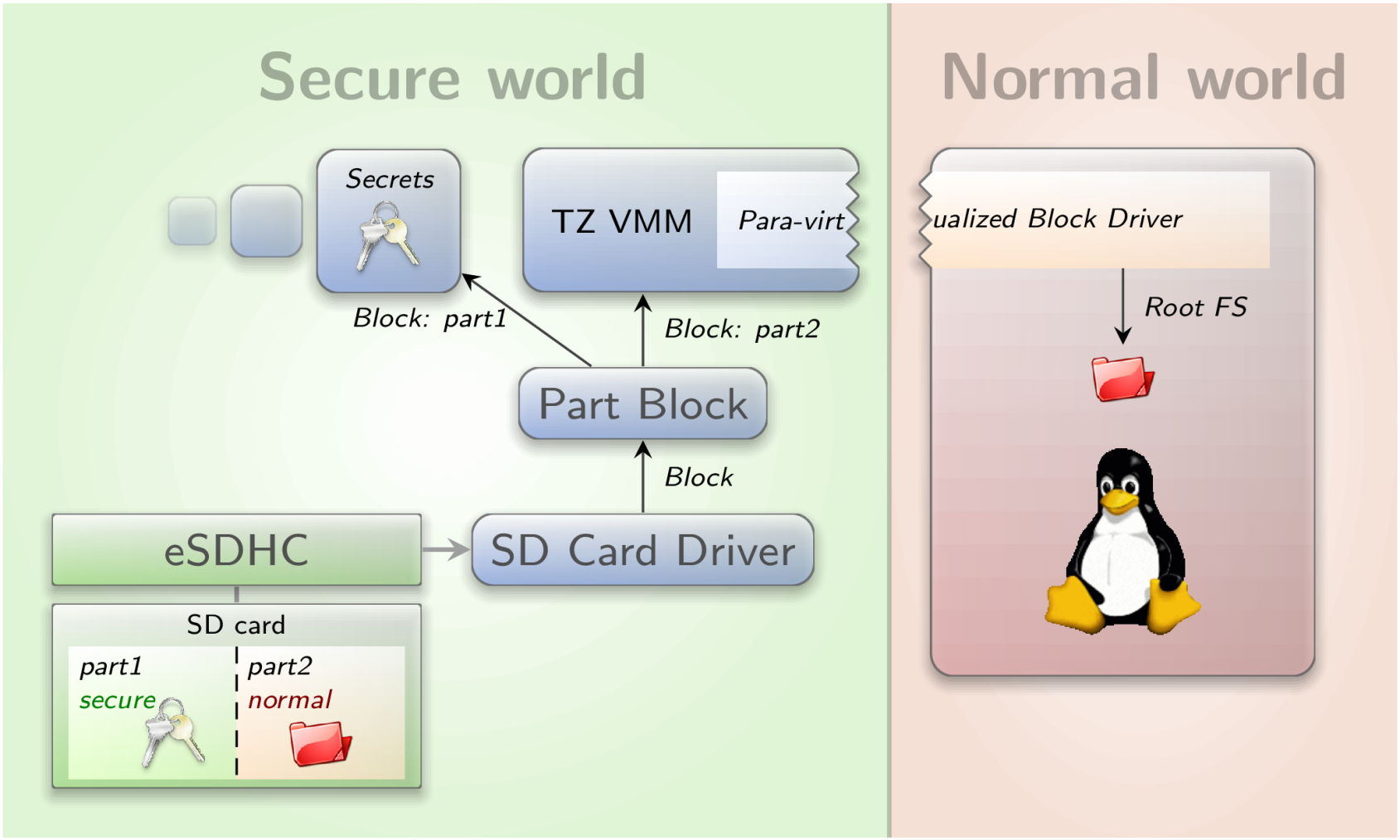
Separation of the SD card



Separation of the SD card



Separation of the SD card



Reliable Output

System got stuck without a hint

Reliable Output

System got stuck without a hint

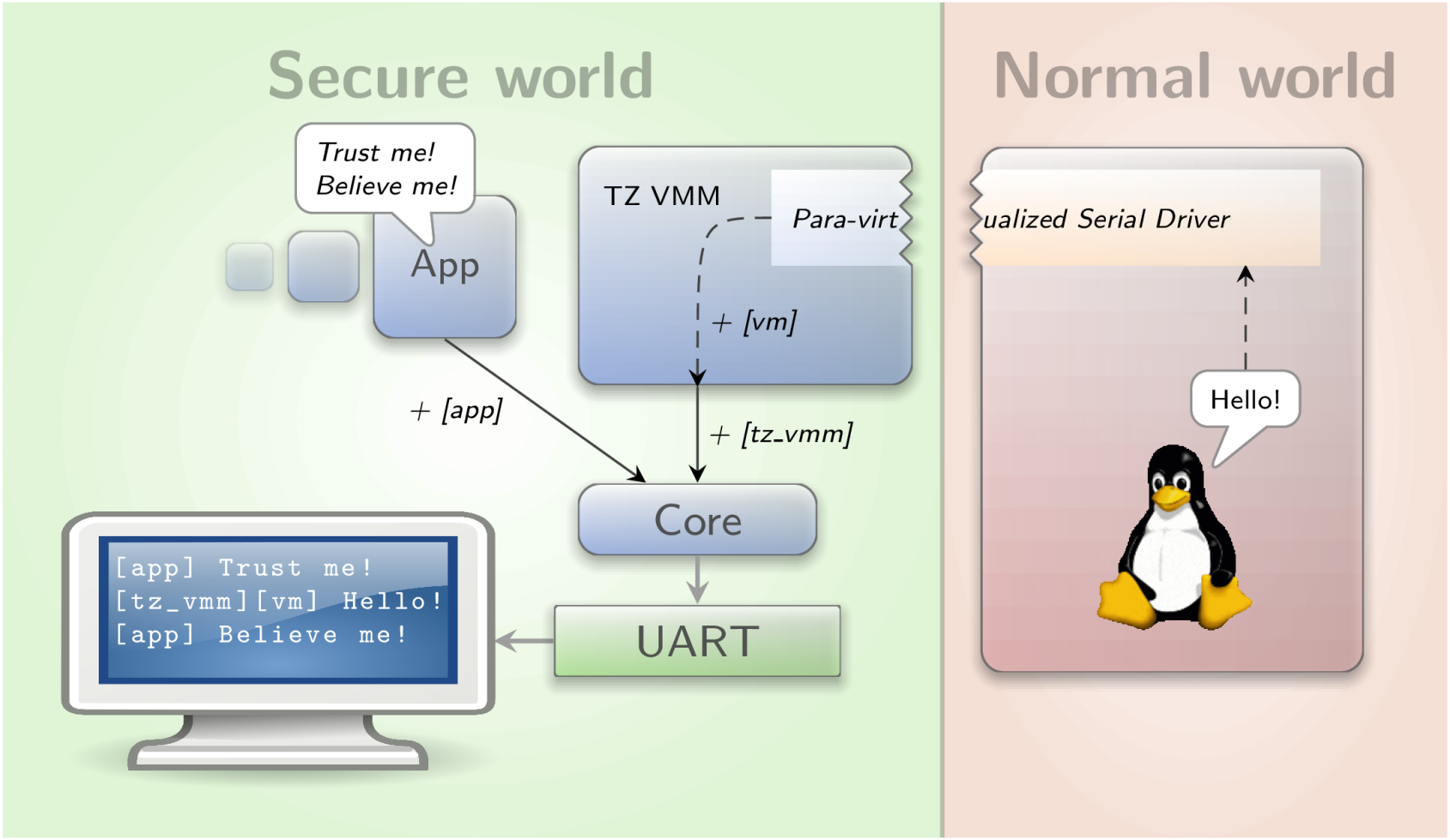
- Triggered LED on world switches and it kept blinking

Reliable Output

System got stuck without a hint

- Triggered LED on world switches and it kept blinking
- Serial output should be para-virtualized too

Reliable Output



Debugging the eSDHC

ESDHC errors after multi-block writes

Debugging the eSDHC

ESDHC errors after multi-block writes

- Checked whether Linux changes GPIO or clocks

Debugging the eSDHC

ESDHC errors after multi-block writes

- Checked whether Linux changes GPIO or clocks
- Reproduced errors in our driver to make them controllable

Debugging the eSDHC

ESDHC errors after multi-block writes

- Checked whether Linux changes GPIO or clocks
- Reproduced errors in our driver to make them controllable
- Tried quirks from eSDHC errata and the new Linux 4.2 sources

Debugging the eSDHC

ESDHC errors after multi-block writes

- Checked whether Linux changes GPIO or clocks
- Reproduced errors in our driver to make them controllable
- Tried quirks from eSDHC errata and the new Linux 4.2 sources
- Checked if Linux got the same errors

Debugging the eSDHC

ESDHC errors after multi-block writes

- Checked whether Linux changes GPIO or clocks
- Reproduced errors in our driver to make them controllable
- Tried quirks from eSDHC errata and the new Linux 4.2 sources
- Checked if Linux got the same errors
- Compared SD commands with new Linux 4.2 trace

Debugging the eSDHC

ESDHC errors after multi-block writes

- Checked whether Linux changes GPIO or clocks
- Reproduced errors in our driver to make them controllable
- Tried quirks from eSDHC errata and the new Linux 4.2 sources
- Checked if Linux got the same errors
- Compared SD commands with new Linux 4.2 trace
  - Linux asks for card state because of flaky host state!

Debugging the eSDHC

ESDHC errors after multi-block writes

- Checked whether Linux changes GPIO or clocks
- Reproduced errors in our driver to make them controllable
- Tried quirks from eSDHC errata and the new Linux 4.2 sources
- Checked if Linux got the same errors
- Compared SD commands with new Linux 4.2 trace
  - Linux asks for card state because of flaky host state!
  - Fixed the errors!

Outline

Features

- Small Genode supervising fully-featured Linux with slight adaptions

Features

- Small Genode supervising fully-featured Linux with slight adaptions
- eSDHC and UART are trustable while controlled Linux access is possible

Features

- Small Genode supervising fully-featured Linux with slight adaptions
- eSDHC and UART are trustable while controlled Linux access is possible
- LED driver

Features

- Small Genode supervising fully-featured Linux with slight adaptions
- eSDHC and UART are trustable while controlled Linux access is possible
- LED driver
- Setup is easy to reproduce and adapt

Open issues

- GPIO and clock controls shared

Open issues

- GPIO and clock controls shared
- LED not trustable

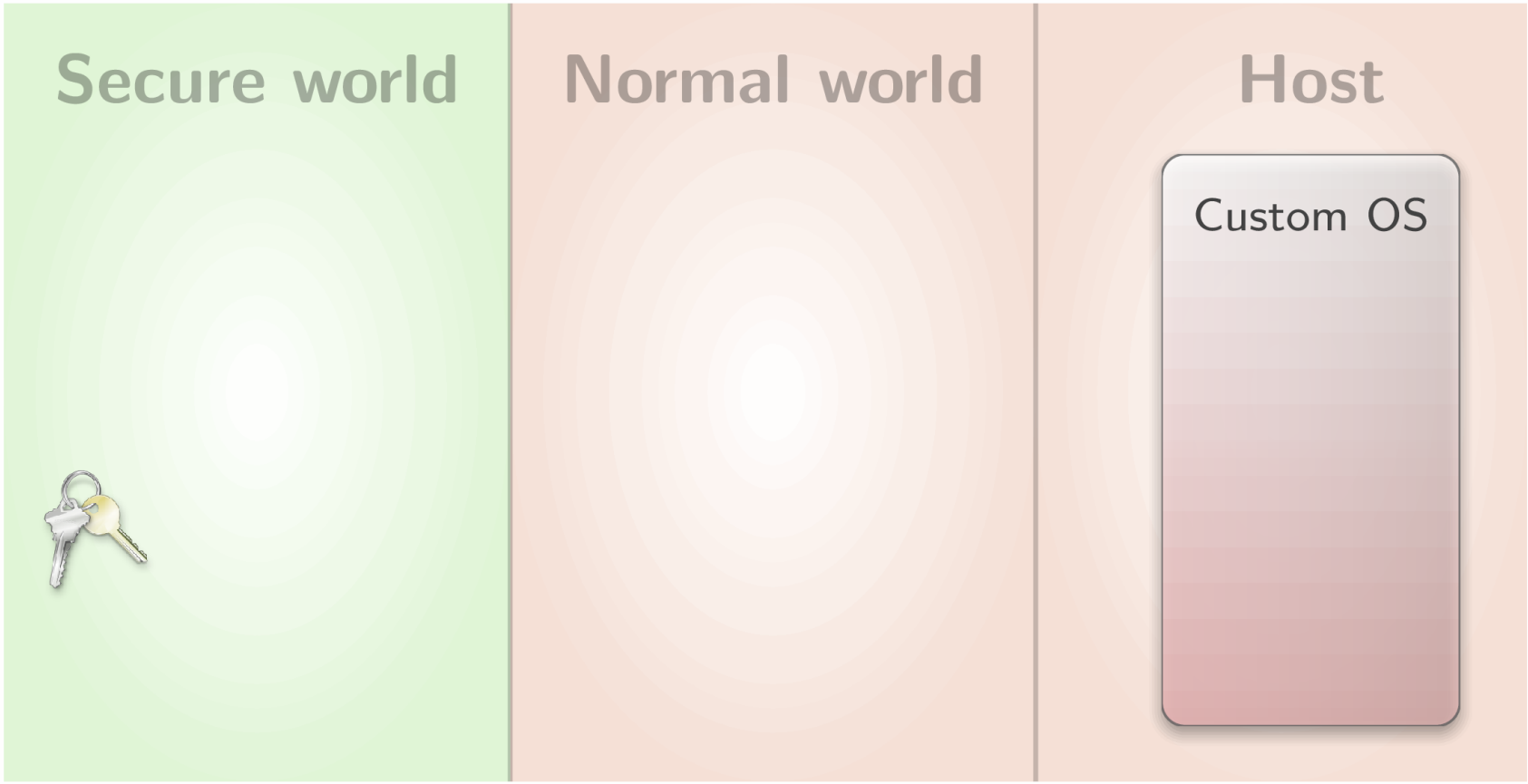
Open issues

- GPIO and clock controls shared
- LED not trustable
- No power saving

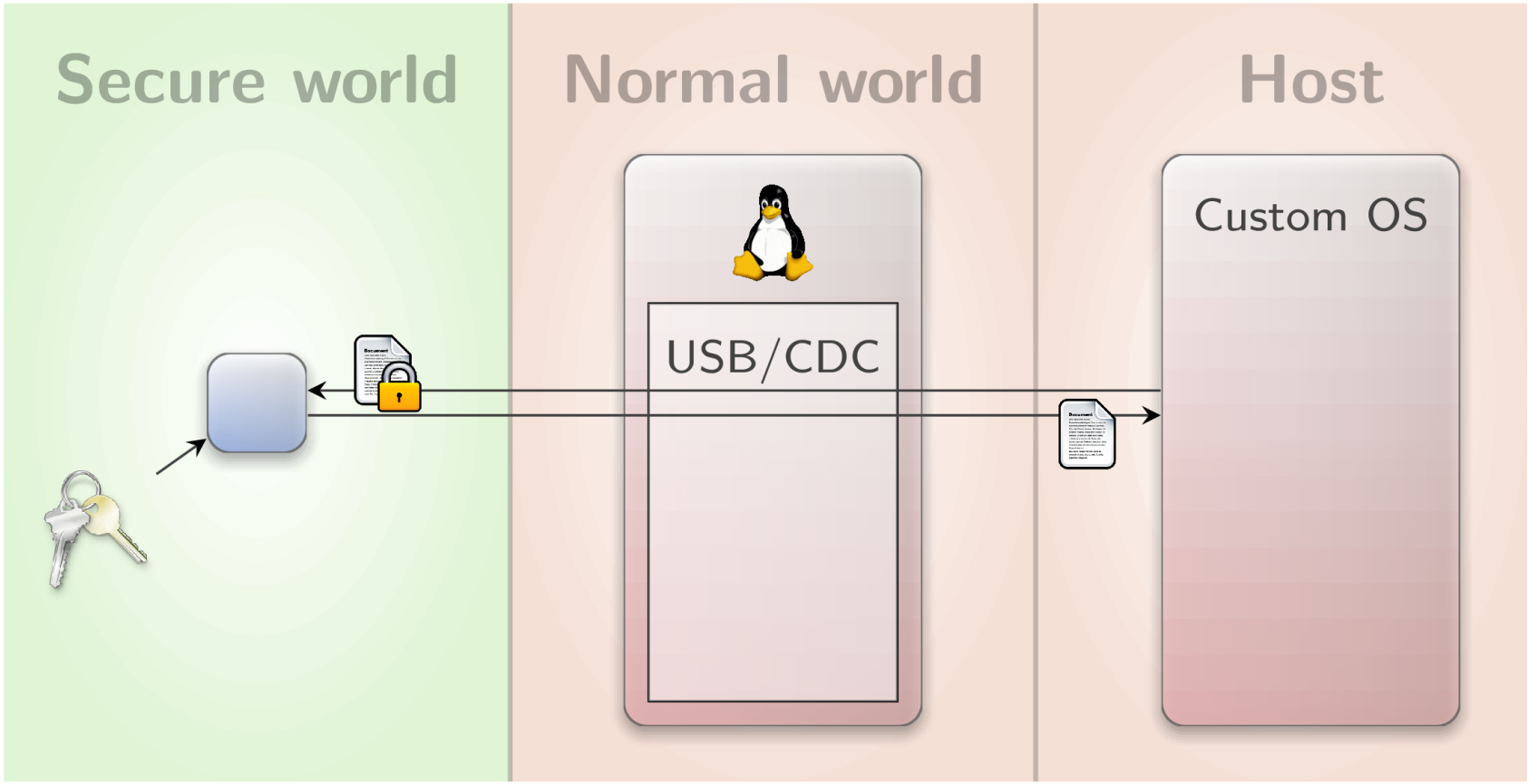
Open issues

- GPIO and clock controls shared
- LED not trustable
- No power saving
- eSDHC driver pretty basic

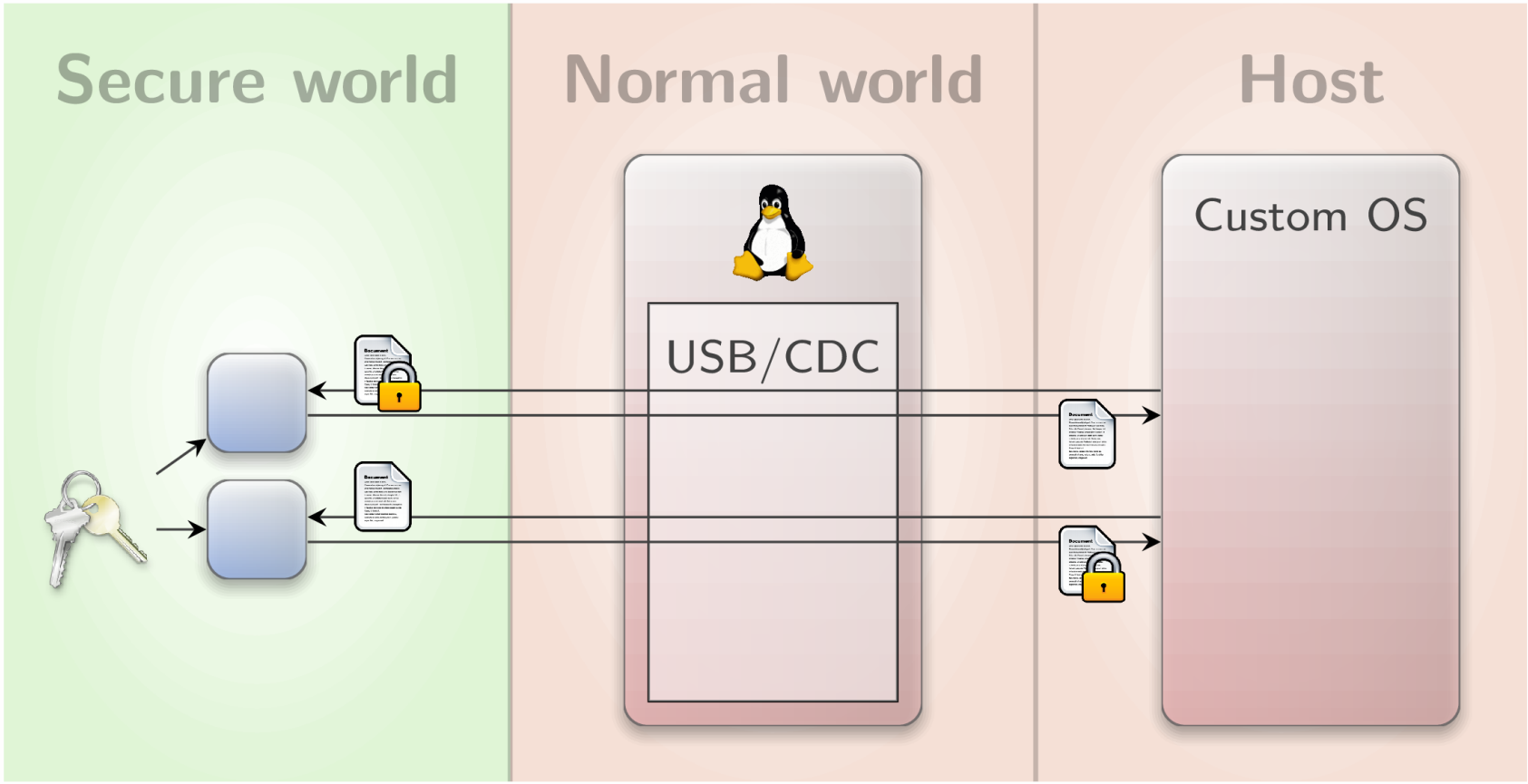
Vision - Protect your secrets



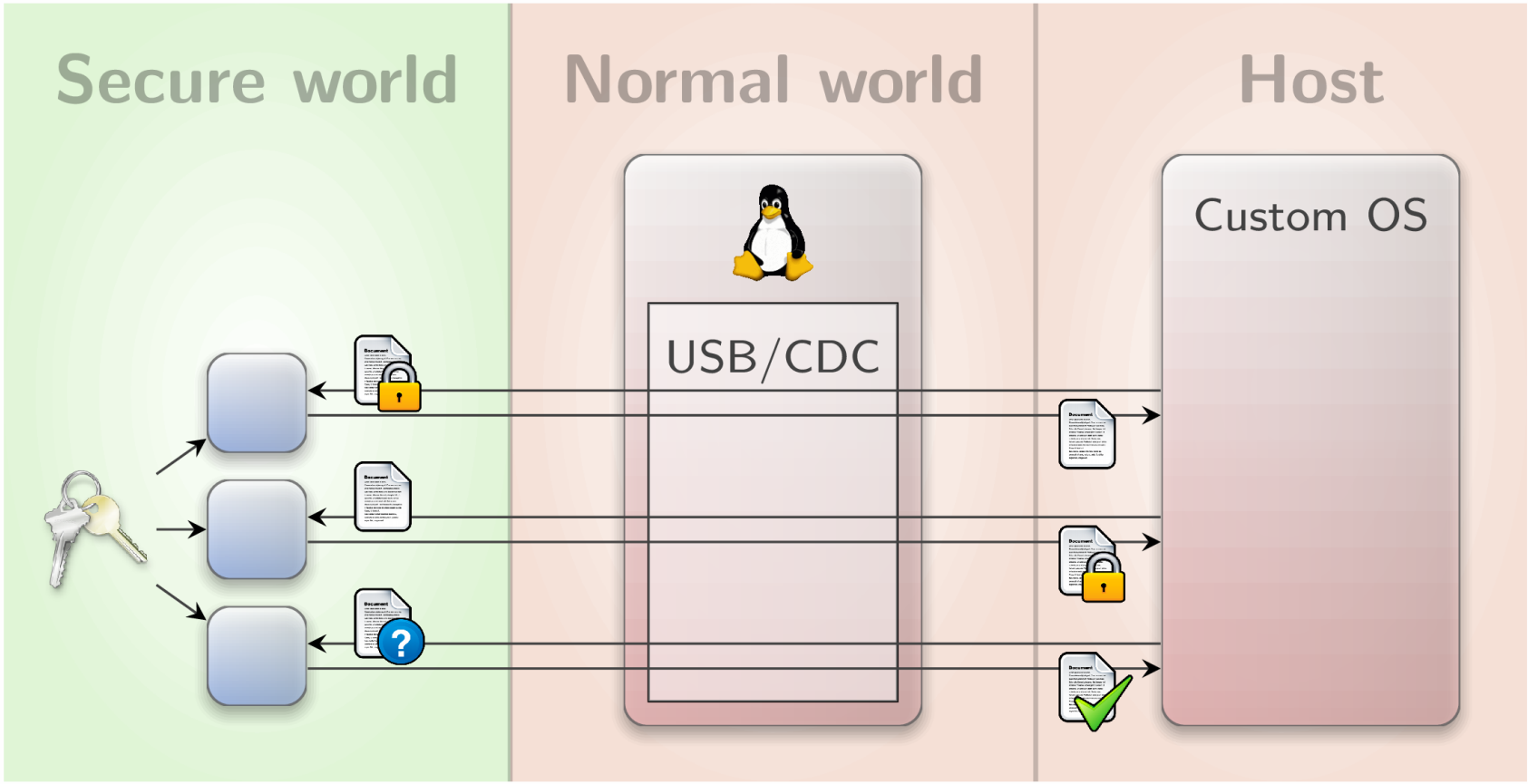
Vision - Protect your secrets



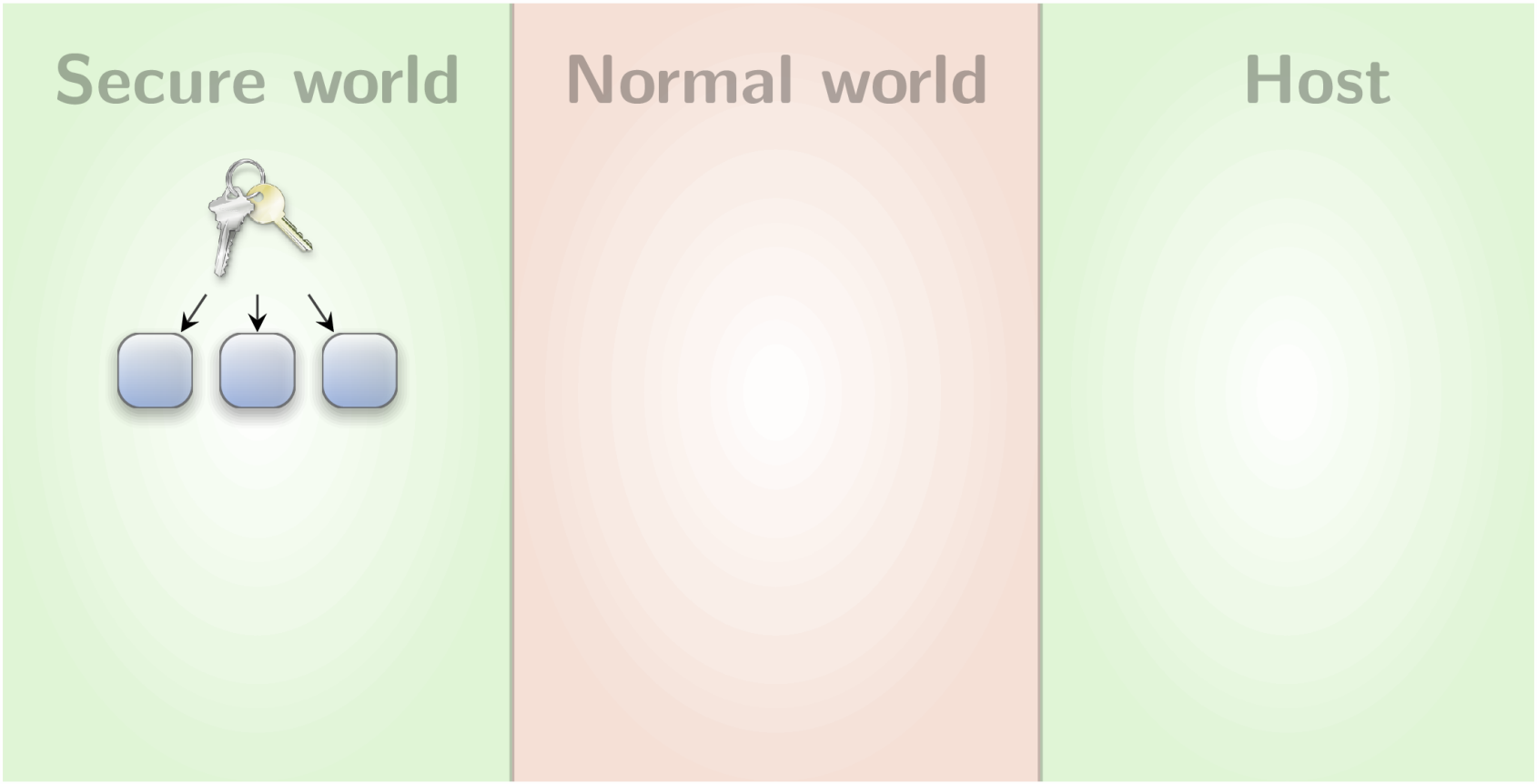
Vision - Protect your secrets



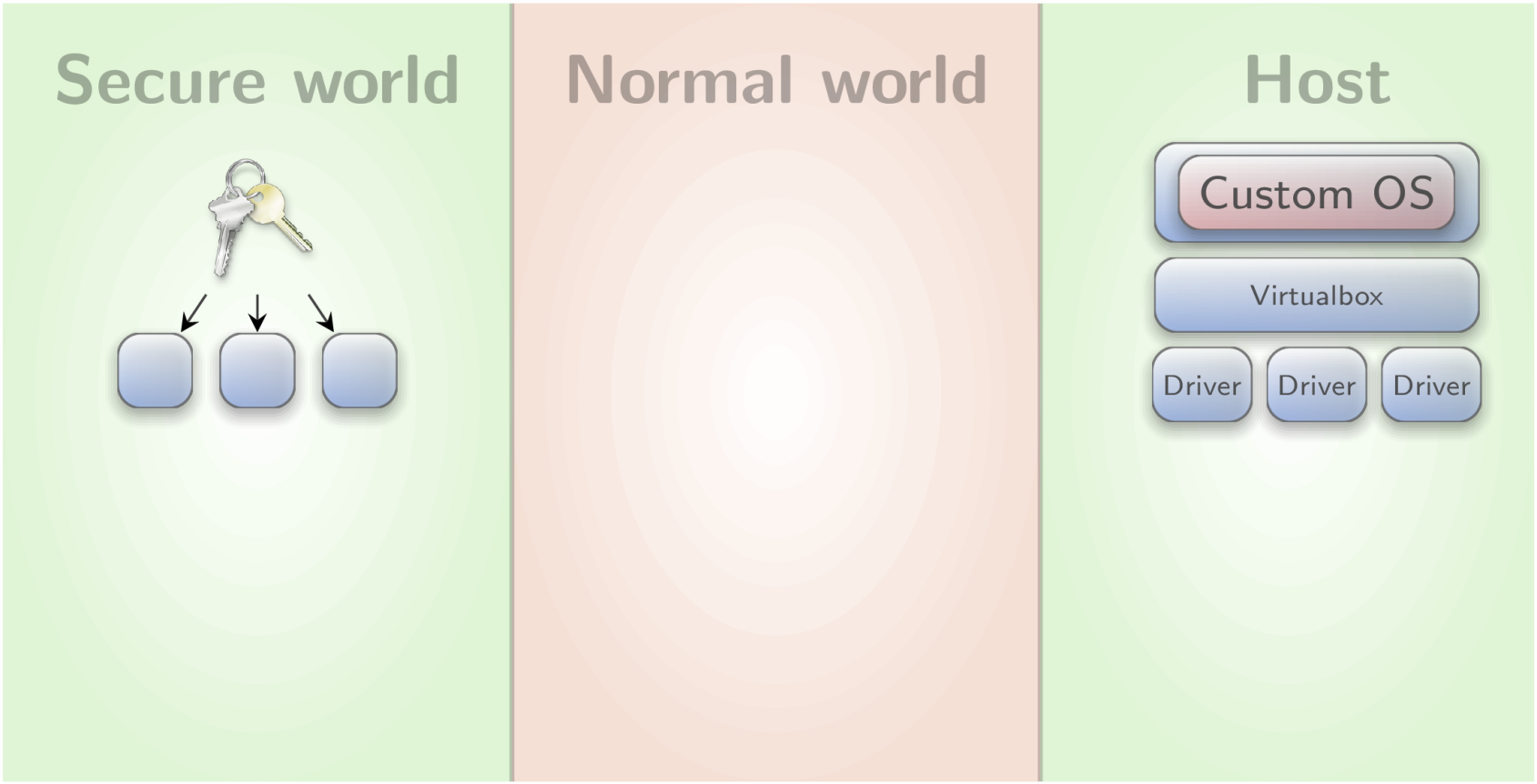
Vision - Protect your secrets



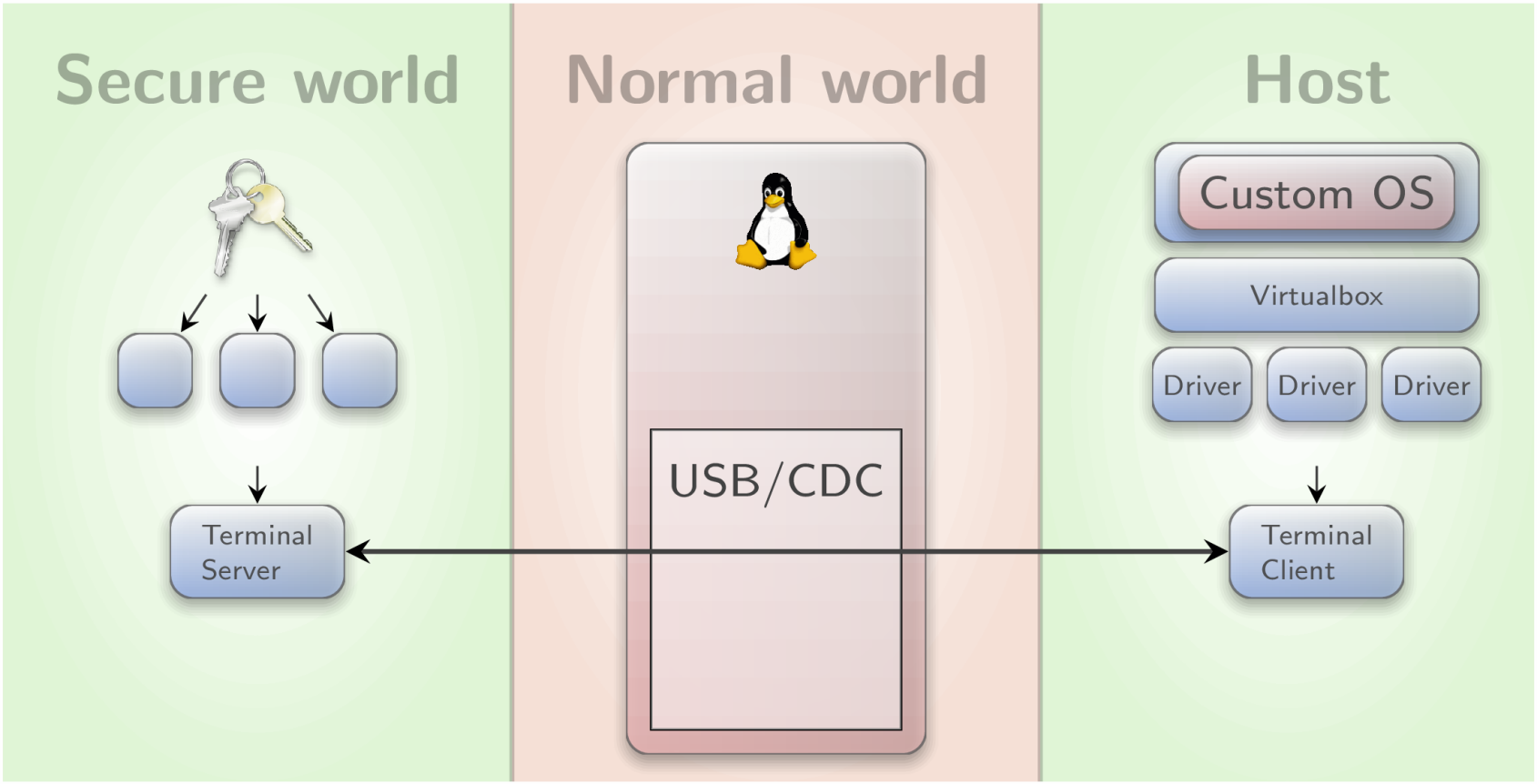
Vision - Distributed Genode



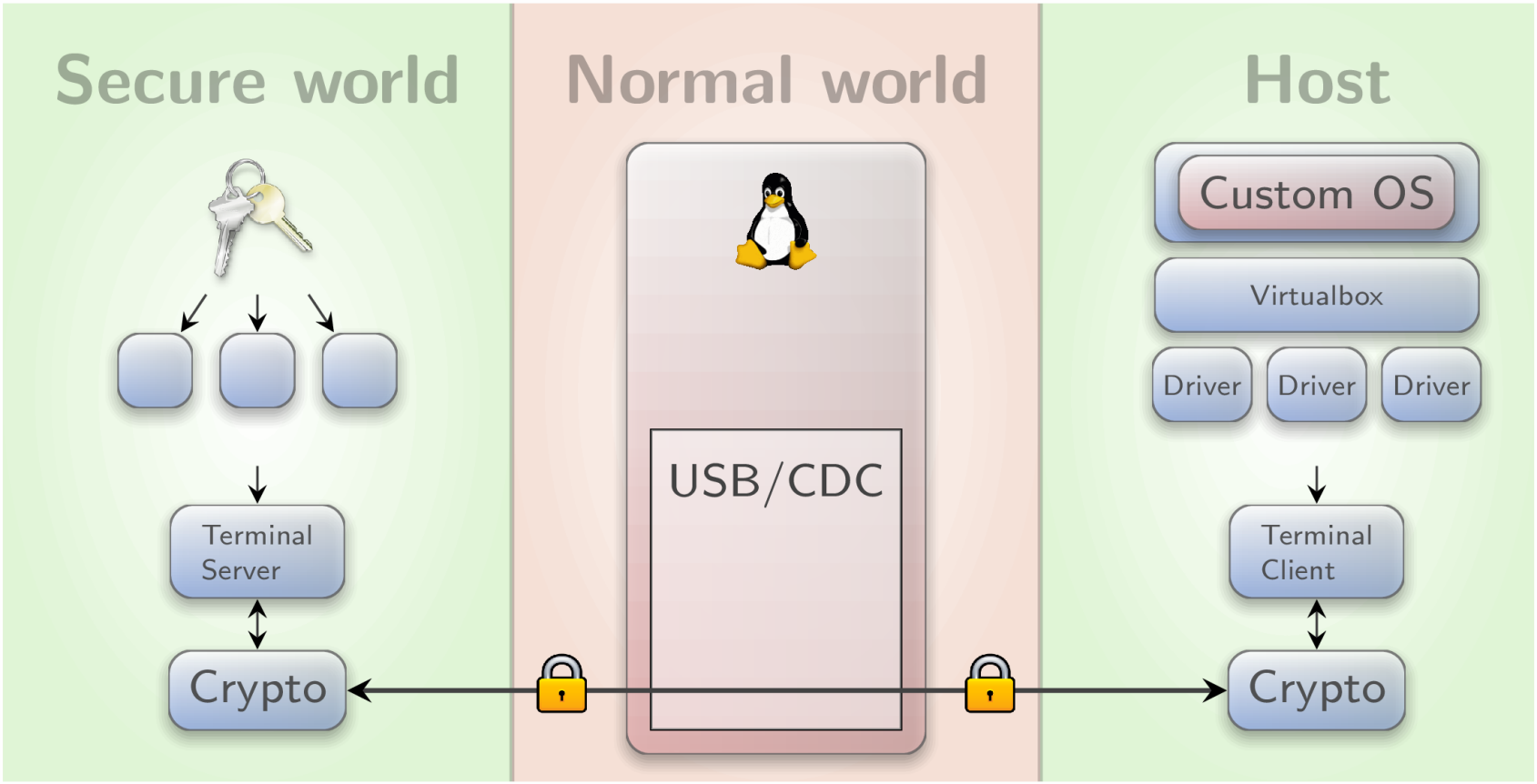
Vision - Distributed Genode



Vision - Distributed Genode



Vision - Distributed Genode



Thank you

USB Armory

`https://inversepath.com/usbarmory`

Genode OS Framework

`http://genode.org`

Genode Labs GmbH

`http://genode-labs.com`