Fusing Hybrid Remote Attestation with a Formally Verified Microkernel: Lessons Learned

Karim Eldefrawy, **Norrathep Rattanavipanon**, Gene Tsudik

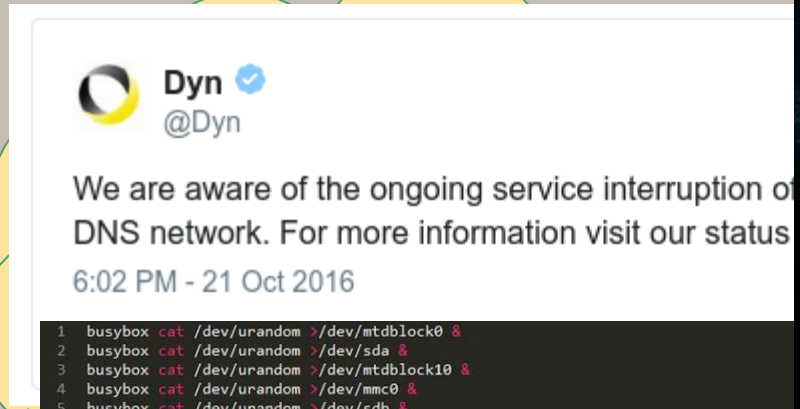
HRL (Currently at SRI) UCI UCI

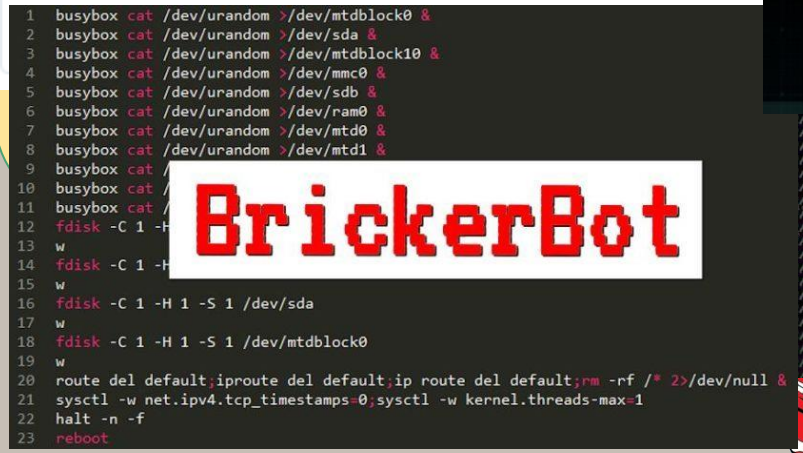
June 28, 2017

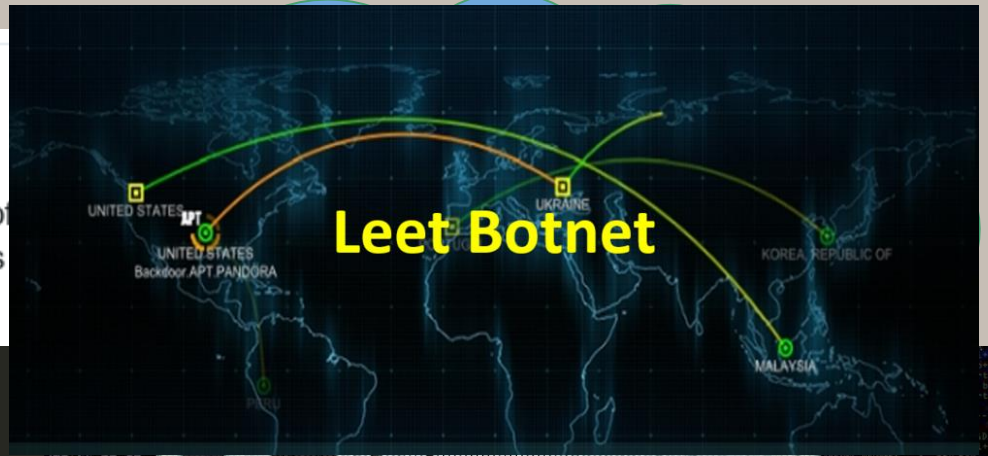
nrattana@uci.edu

http://sprout.ics.uci.edu

IoT/CPS/ES


Dyn
@Dyn
We are aware of the ongoing service interruption of
DNS network. For more information visit our status
6:02 PM - 21 Oct 2016

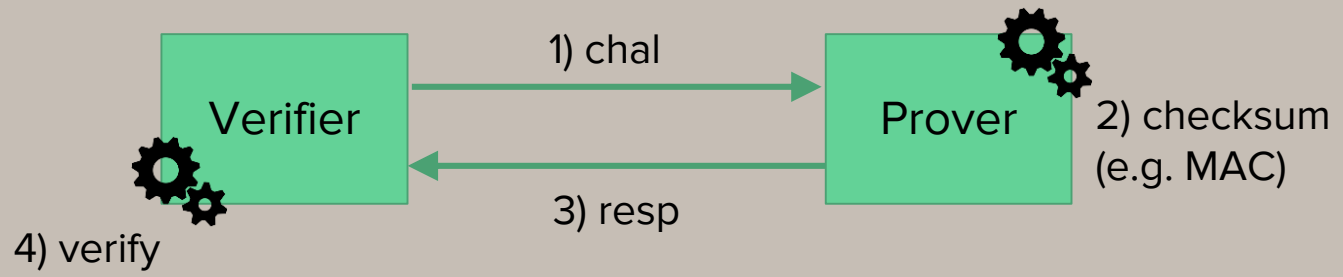

1 busybox cat /dev/urandom >/dev/mtdblock0 &
2 busybox cat /dev/urandom >/dev/sda &
3 busybox cat /dev/urandom >/dev/mtdblock10 &
4 busybox cat /dev/urandom >/dev/mmc0 &
5 busybox cat /dev/urandom >/dev/sdb &
6 busybox cat /dev/urandom >/dev/ram0 &
7 busybox cat /dev/urandom >/dev/mtd0 &
8 busybox cat /dev/urandom >/dev/mtd1 &
12 fdisk -C 1 -H
13 w
14 fdisk -C 1 -H
15 w
16 fdisk -C 1 -H 1 -S 1 /dev/sda
17 w
18 fdisk -C 1 -H 1 -S 1 /dev/mtdblock0
19 w
20 route del default;iproute del default;ip route del default;rm -rf /* 2>/dev/null &
21 sysctl -w net.ipv4.tcp_timestamps=0;sysctl -w kernel.threads-max=1
22 halt -n -f
23 reboot
BrickerBot


Leet Botnet
UNITED STATES
UNITED STATES
Backdoor APT PANDORA
UKRAINE
KOREA, REPUBLIC OF
MALAYSIA


StuXnet

Remote Attestation (RA)

- ★ Remote verification of internal state of a prover by a verifier
  - Secure updates, deletion/erasure and resetting
- ★ Challenge-response protocol between
  - Trusted verifier : powerful entity
  - Untrusted prover : embedded device



Design of RA

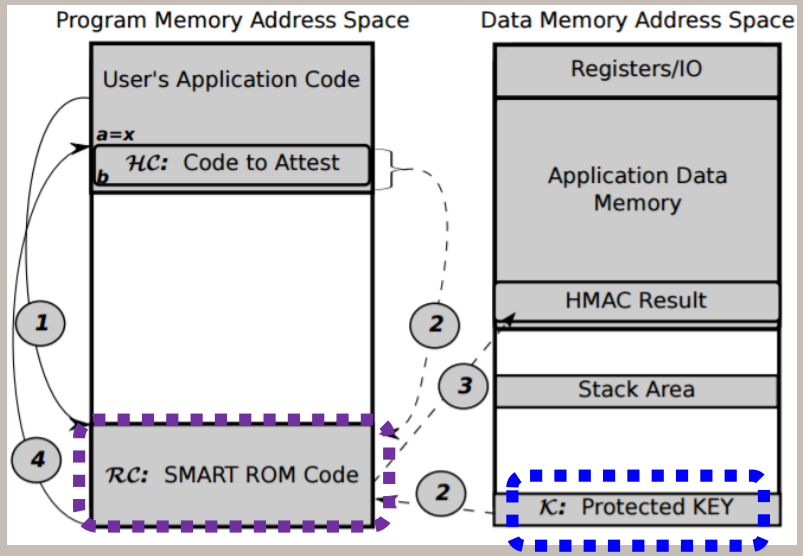
- ★ Hardware-only Attestation
  - Secure hardware (e.g. TPM)
  - Overkill for medium/low-end IoT/embedded devices
- ★ Software-only Attestation
  - A.k.a. timing-based attestation
  - Does not support multi-hop communication
  - Underlying assumptions (seriously) challenged [1]
- ★ Hybrid Attestation
  - Minimal hardware support for secure RA

[1] C. Castellucia, et al. On the difficulty of Software-Based Attestation of Embedded Devices, CCS 2009.

SMART

First Hybrid Design in Remote Attestation for low-end microcontrollers (MCUs)

SMART

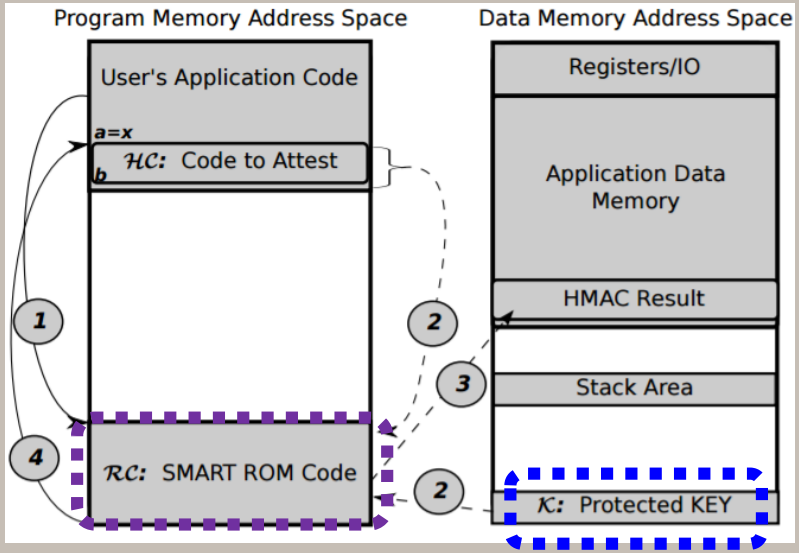


Properties

1) Exclusive Access to Key
2) No Leaks
3) Immutability
4) Uninterruptability
5) Controlled Invocation

K. Eldefrawy, et al. Secure & Minimal Architecture for Remote Trust, NDSS 2012.
A. Francillon, et al. A Minimalist Approach to Remote Attestation, DATE 2014.

SMART



Properties

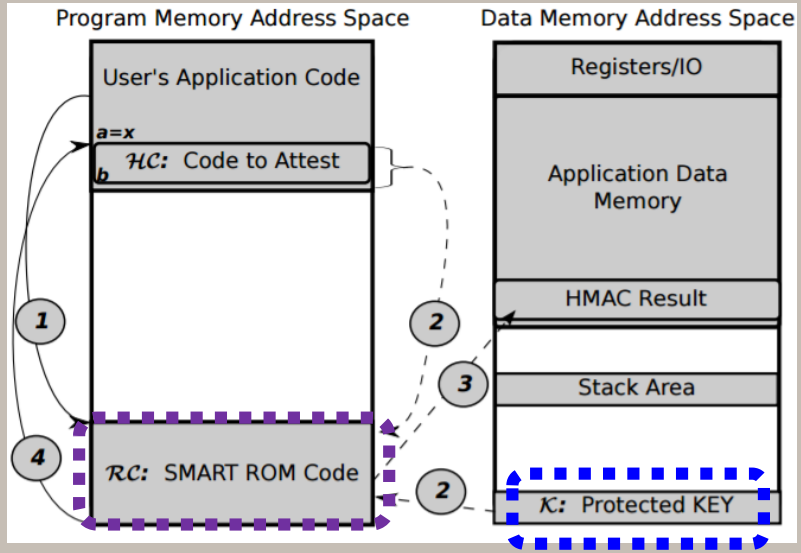
1) Exclusive Access to Key
2) No Leaks
3) Immutability
4) Uninterruptability
5) Controlled Invocation

Hardware Requirement

★ ROM

★ MCU (bus) access controls

SMART



Properties

1) Exclusive Access to Key
2) No Leaks
3) Immutability
4) Uninterruptability
5) Controlled Invocation

Hardware Requirement

★ ROM

★ MCU (bus) access controls

Can be emulated using a formally verified software component

Fusing Hybrid RA Design with seL4

seL4

What is it?

- Formal verification of the kernel
  - Spec ➡ Impl ⇨ Binary
- Capability-based access control
- Formally proven access control enforcement

seL4

What is it?

- Formal verification of the kernel
  - Spec → Impl → Binary
- Capability-based access control
- Formally proven access control enforcement

Why?

- Emulate hardware-enforced access controls in SMART
- Provide isolation of Attestation Process

seL4

What is it?

- Formal verification of the kernel
  - Spec → Impl → Binary
- Capability-based access control
- Formally proven access control enforcement

Why?

- Emulate hardware-enforced access controls in SMART
- Provide isolation of Attestation Process

How?

Map SMART properties into seL4 configuration

Deriving Configuration

SMART Properties

- ★ Exclusive Access (E/A) to key
- ★ No leaks
- ★ Immutability
- ★ Uninterruptability
- ★ Controlled invocation

Att Process Config.

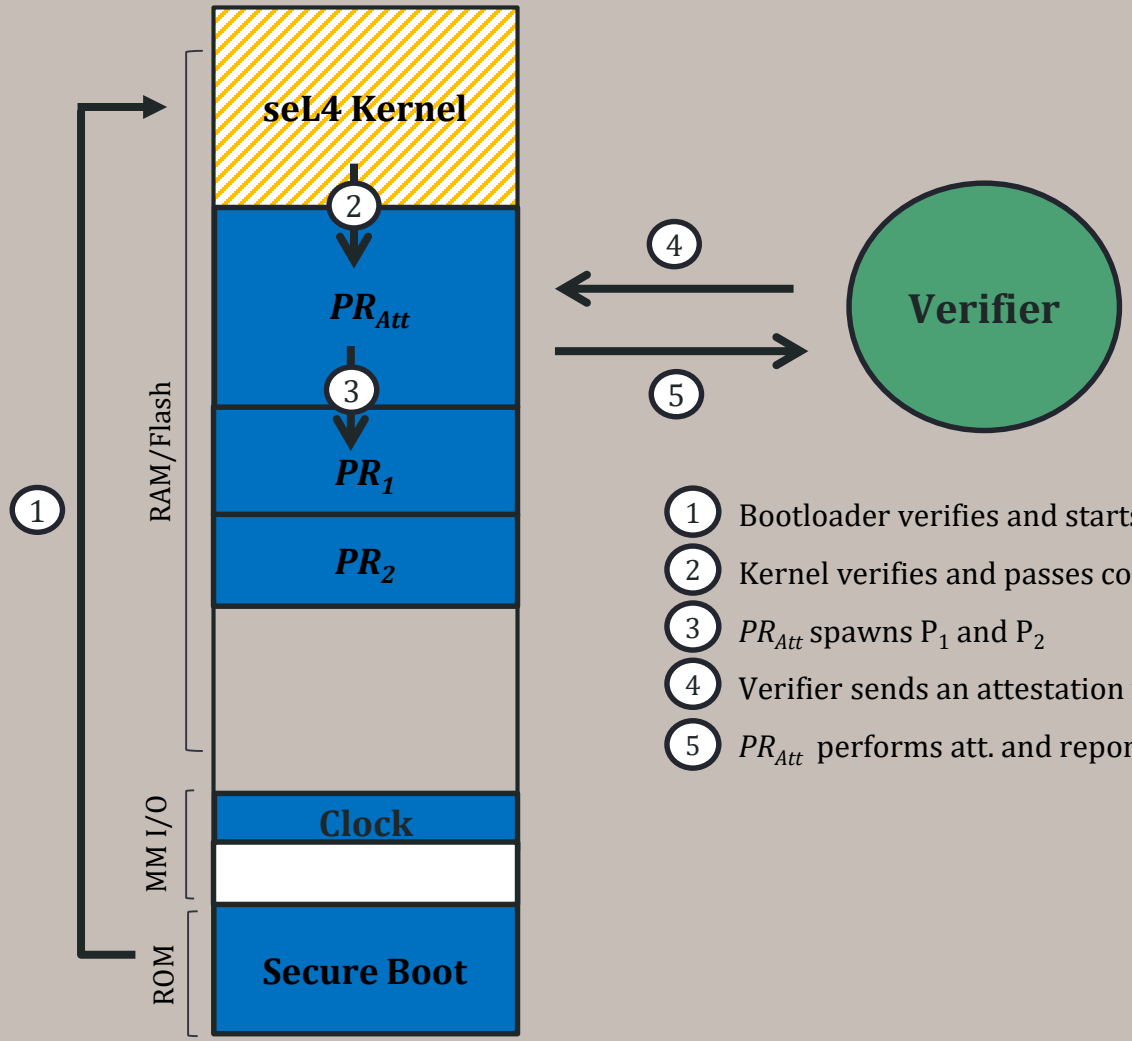
- ★ E/A to key
- ★ E/A to virtual space
- ★ E/A to executable
- ★ Secure boot of seL4 and att. process
- ★ Highest priority
- ★ E/A to Thread Control Block (TCB)

Our Approach - HYDRA

- ★ Run Attestation Process (PR_{Att}) as *initial user-space process*
  - Contains capabilities to all objects, e.g. IPC, page and thread
  - Runs with highest scheduling priority
  - Manages the rest of user-space

Our Approach - HYDRA

- ★ Run Attestation Process (PR_{Att}) as *initial user-space process*
  - Contains capabilities to all objects, e.g. IPC, page and thread
  - Runs with highest scheduling priority
  - Manages the rest of user-space
- ★ Only spawn new process that does not contain capabilities to PR_{Att}'s:
  - Executable/Key
  - Working virtual memory
  - TCB



1. Bootloader verifies and starts seL4 microkernel
2. Kernel verifies and passes control to PR_{Att}
3. PR_{Att} spawns P_1 and P_2
4. Verifier sends an attestation request to PR_{Att}
5. PR_{Att} performs att. and reports back to verifier

Implementation

- Prototype on **I.MX6-SabreLite**

https://boundarydevices.com/product/sabre-lite-imx6-sbc/

- Existing secure boot mechanism in Sabre Lite

Challenges / Lessons Learned

Challenges when using seL4

Formal verification of seL4 assumes:

- ★ Proper initialization of the kernel
  - Motivate using hardware-enforced secure boot

Challenges when using seL4

Formal verification of seL4 assumes:

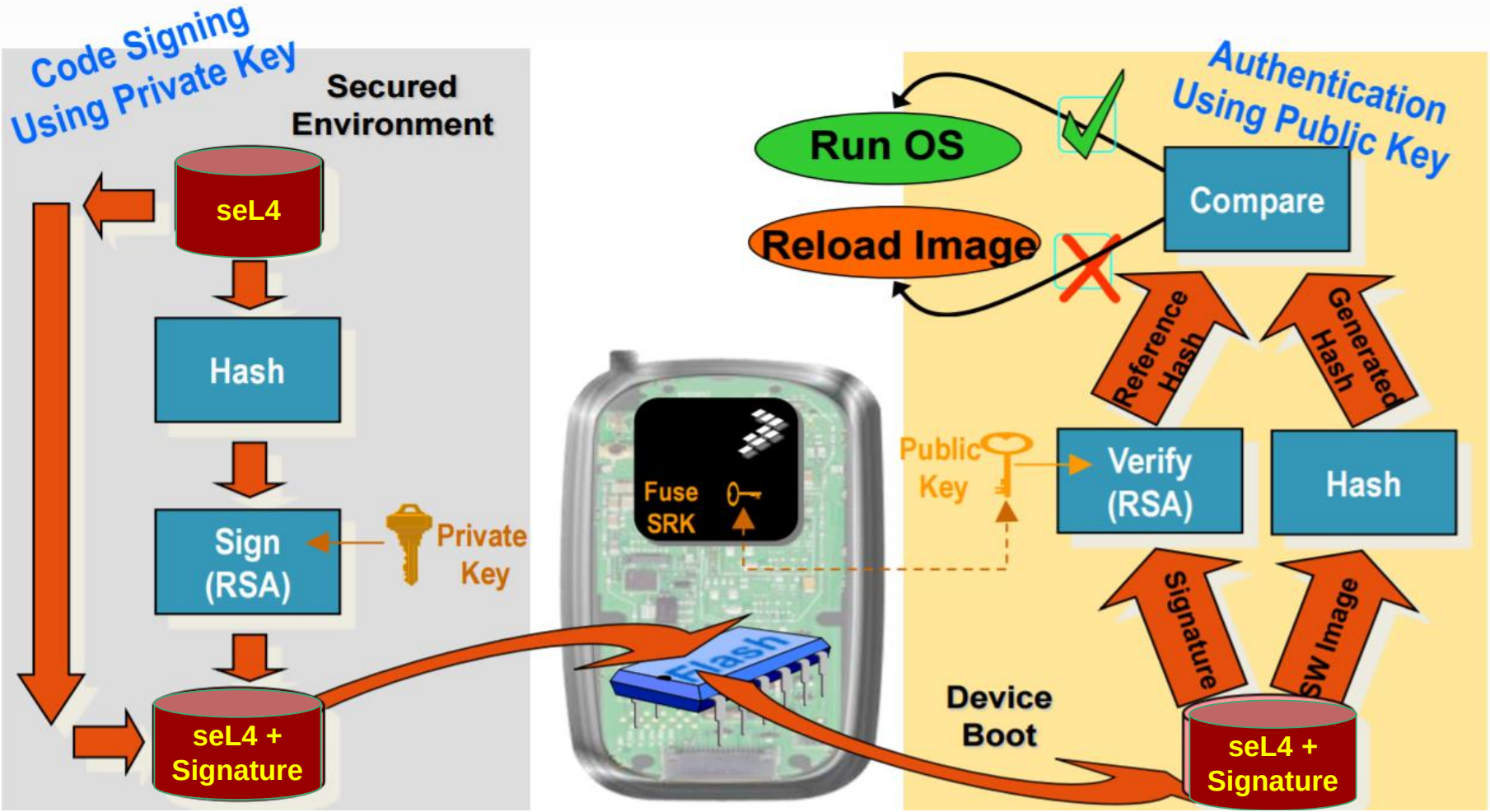
- ★ Proper initialization of the kernel
  - Motivate using hardware-enforced secure boot
- ★ Correct behavior of the underlying hardware
  - Sol: Run seL4 on top of a formally verified processor
  - But does such hardware exist?
  - Not yet ... but possible in the future, e.g. CHERI ISA [1] based on Bluespec SystemVerilog [2]

[1] R. N. Watson, et al. Capability hardware enhanced risc instructions: Cheri instruction-set architecture, 2016.
[2] R. Nikhil and K. Czeck, BSV by Example. CreateSpace Independent Publishing Platform, 2010

Platform Specific Secure Boot

- ★ Requires configurable/programmable secure boot
  - Not easy to find in commercial development boards

- ★ SabreLite's High Assurance Boot (HAB)
  - Based on a digital signature scheme
  - Configurable through ROM APIs



Rod Ziolkowski, i.MX Applications Processor Trust Architecture, 2013

Ensuring Freshness of Attestation Requests

- ★ Computational Denial-of-Service from:
  - Bogus requests
  - Delay, replay or reordering attacks
- ★ Solution: Verify requests!
  - Requires timestamp generated by a reliable read-only clock.
  - Read-only property can be enforced using seL4's capability.
  - Reliable property requires a (semi-synchronous) real-time clock.

F. Brasser, et al. Remote Attestation for Low-End Embedded Devices: the Prover's Perspective, DAC 2016.

Ensuring Freshness of Attestation Requests

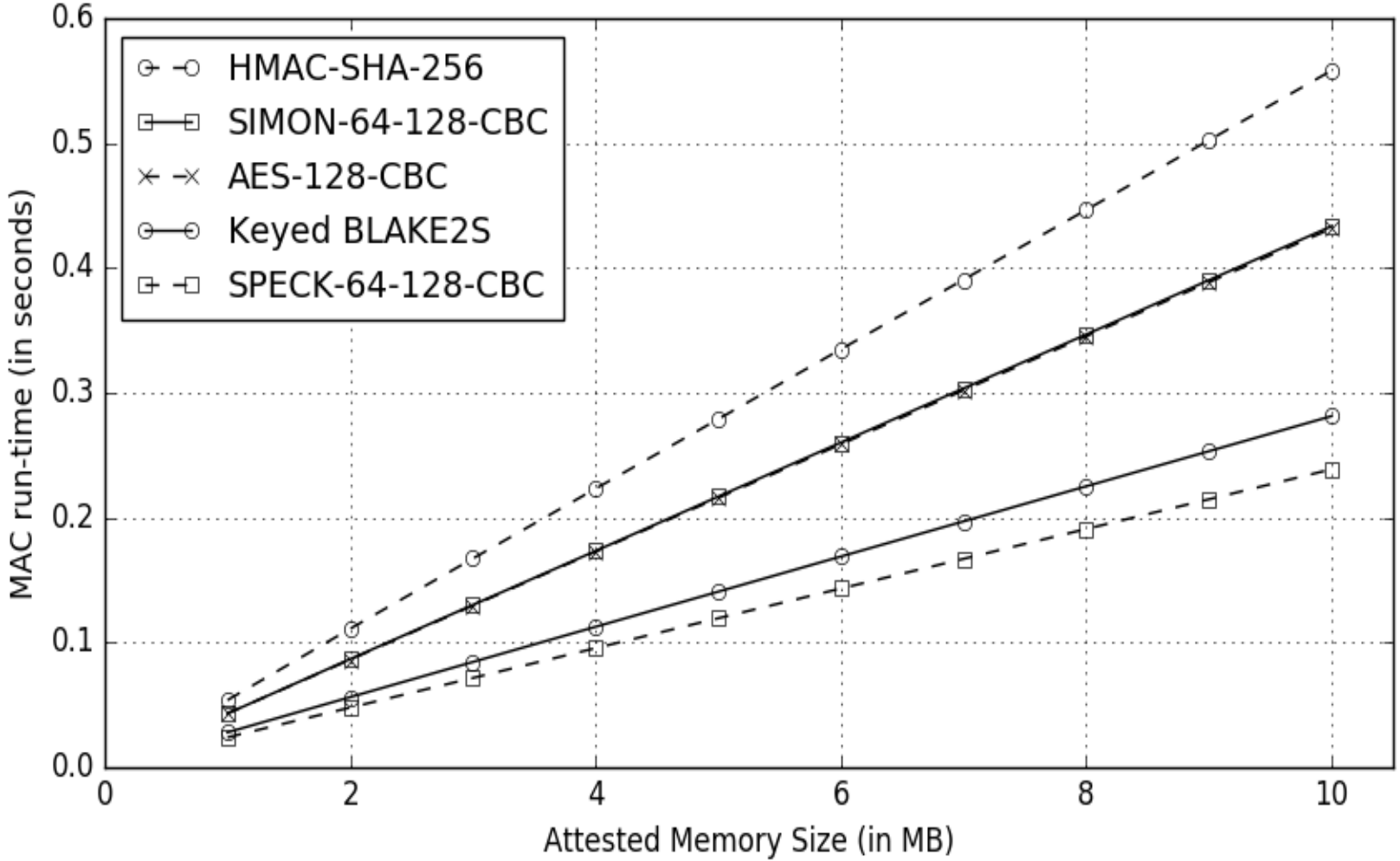
- ★ No real-time clock driver implementation in Sabre Lite.
- ★ Workaround by generating pseudo-timestamp using a counter + a secondary storage
  - When PR_{Att} starts, it loads T_0 that was saved before the last reboot.
  - When the **first** request arrives, check its timestamp (T_1) with T_0
  - Verify request. If success, keep track of T_1 and start counter.
  - $TS = T_1$ + counter value
  - Periodically store TS

Conclusion

- First hybrid design for Remote Attestation using a formally verified microkernel (seL4)
  - Emulate certain properties that were previously only realizable using hardware features
- Prototype on commercially available platform
- Challenges
  - seL4 assumptions
  - Secure boot
  - Timestamp generation

Questions?

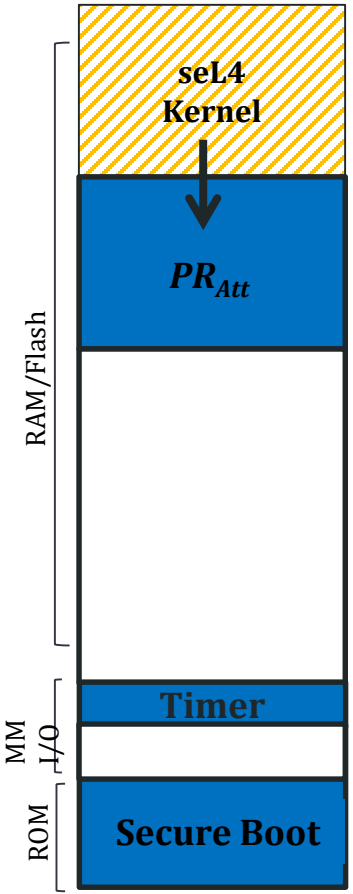
References

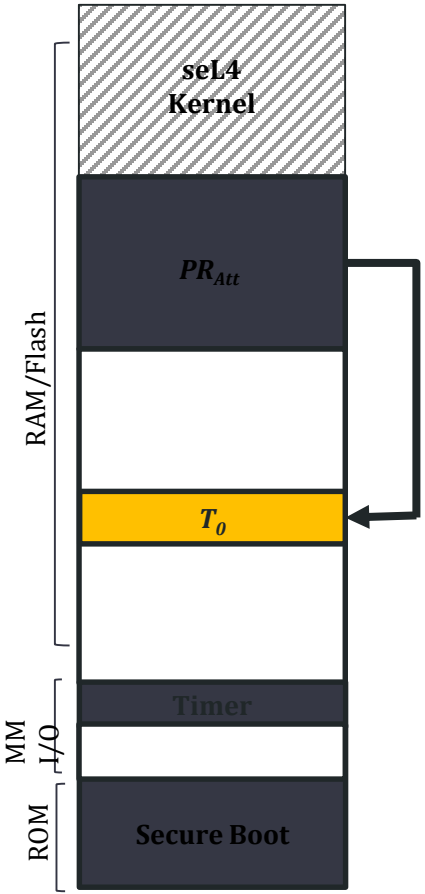

HMAC-SHA-256
SIMON-64-128-CBC
AES-128-CBC
Keyed BLAKE2S
SPECK-64-128-CBC
MAC run-time (in seconds)
Attested Memory Size (in MB)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
2
4
6
8
10

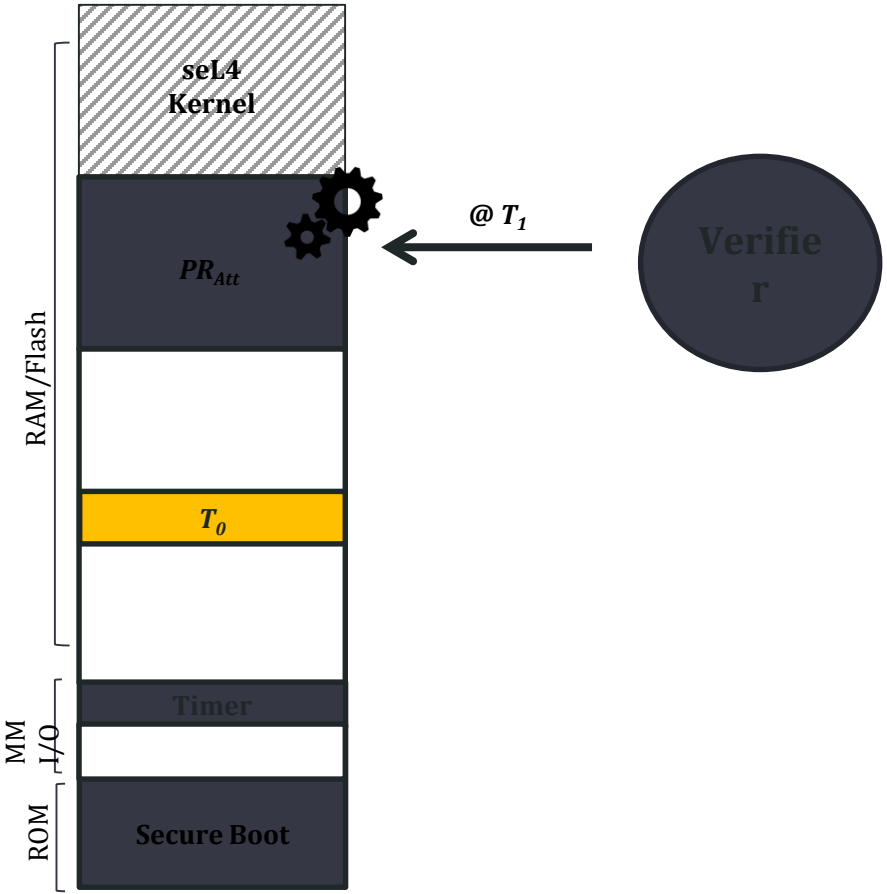
Ensuring Freshness of Attestation Requests

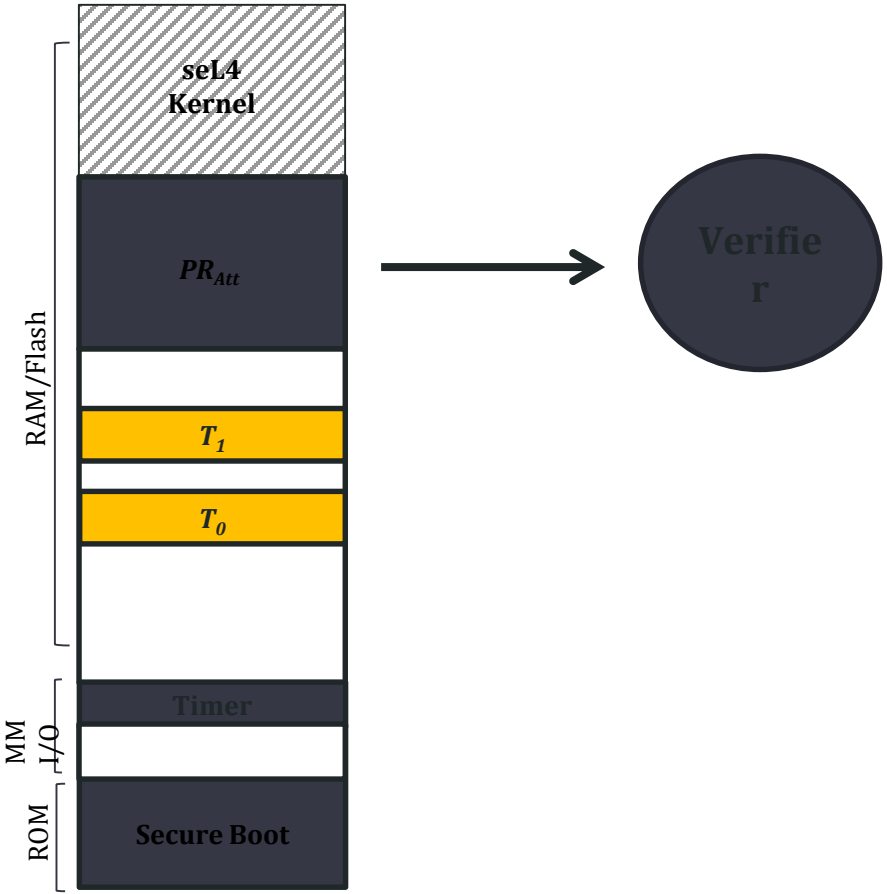
- ★ No real-time clock driver implementation in Sabre Lite.

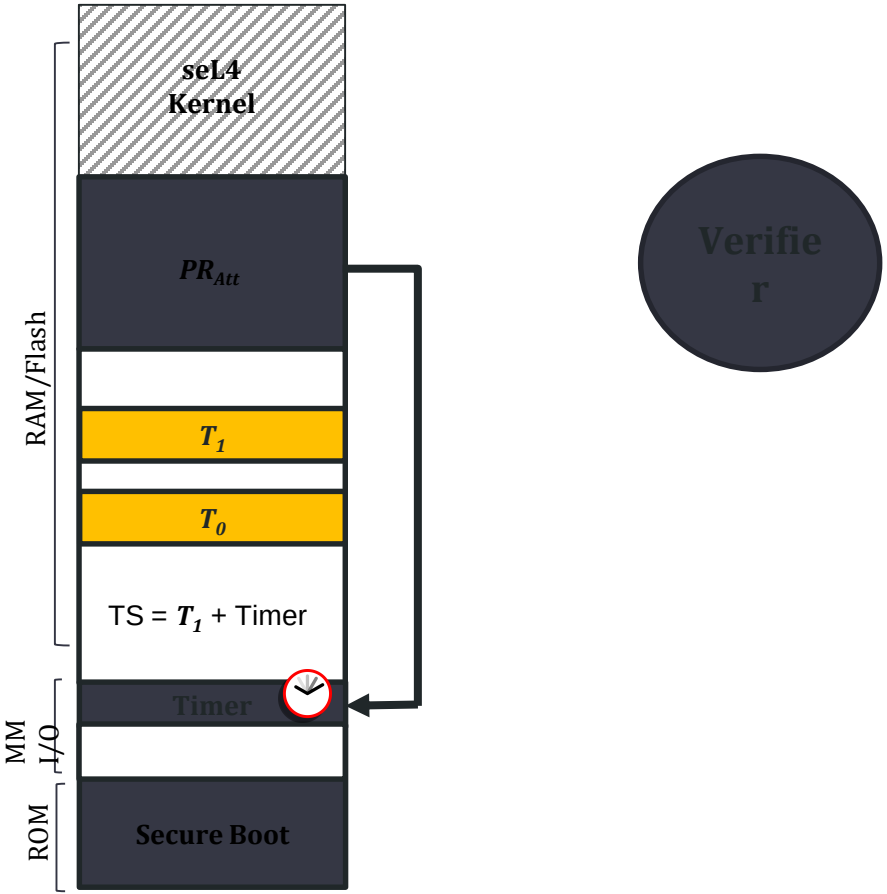
- ★ Workaround by generating pseudo-timestamp using a counter + a secondary storage


seL4
Kernel
PR_{Att}
RAM/Flash
Timer
MM
I/O
Secure Boot
ROM


seL4
Kernel
PR_{Att}
T_0
Timer
Secure Boot
RAM/Flash
MM
I/O
ROM


seL4
Kernel
PR_Att
@ T_1
Verifie
r
RAM/Flash
T_0
Timer
MM
I/O
ROM
Secure Boot

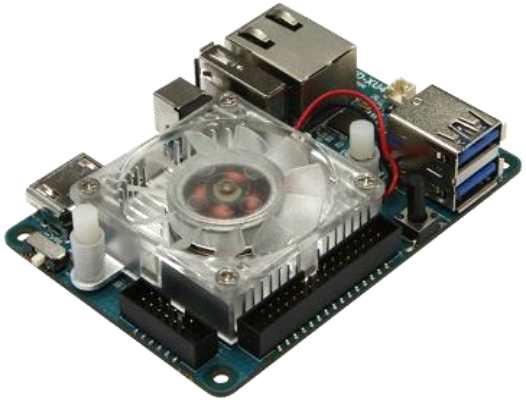

seL4
Kernel
PR_{Att}
T_1
T_0
Timer
Secure Boot
RAM/Flash
MM
I/O
ROM
Verifie
r


seL4
Kernel
PR_{Att}
T_1
T_0
TS = T_1 + Timer
Timer
Secure Boot
RAM/Flash
MM
I/O
ROM
Verifie
r

Implementation

- Prototype on **I.MX6-SabreLite** and **Odroid-XU4**

https://boundarydevices.com/product/sabre-lite-imx6-sbc/

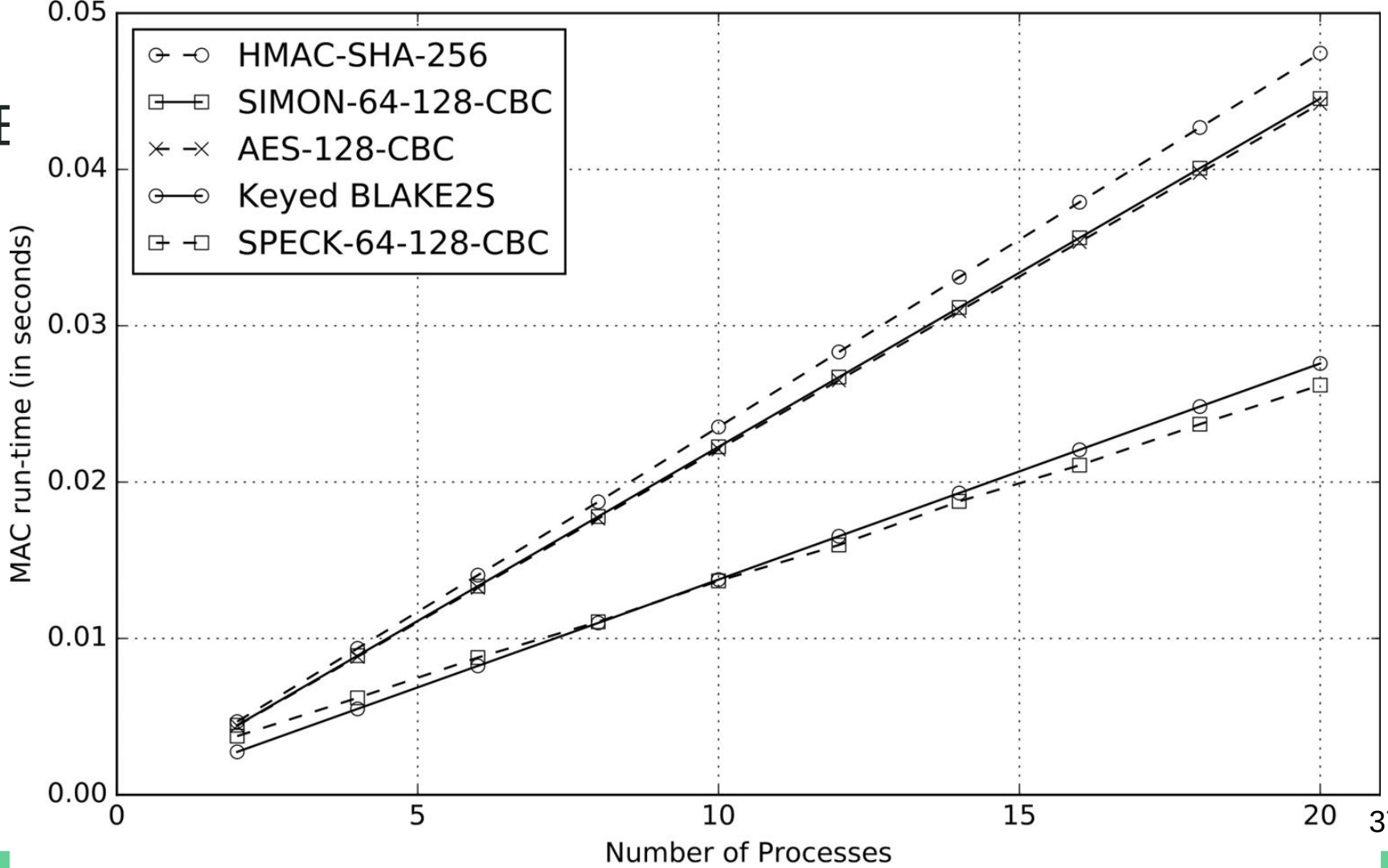


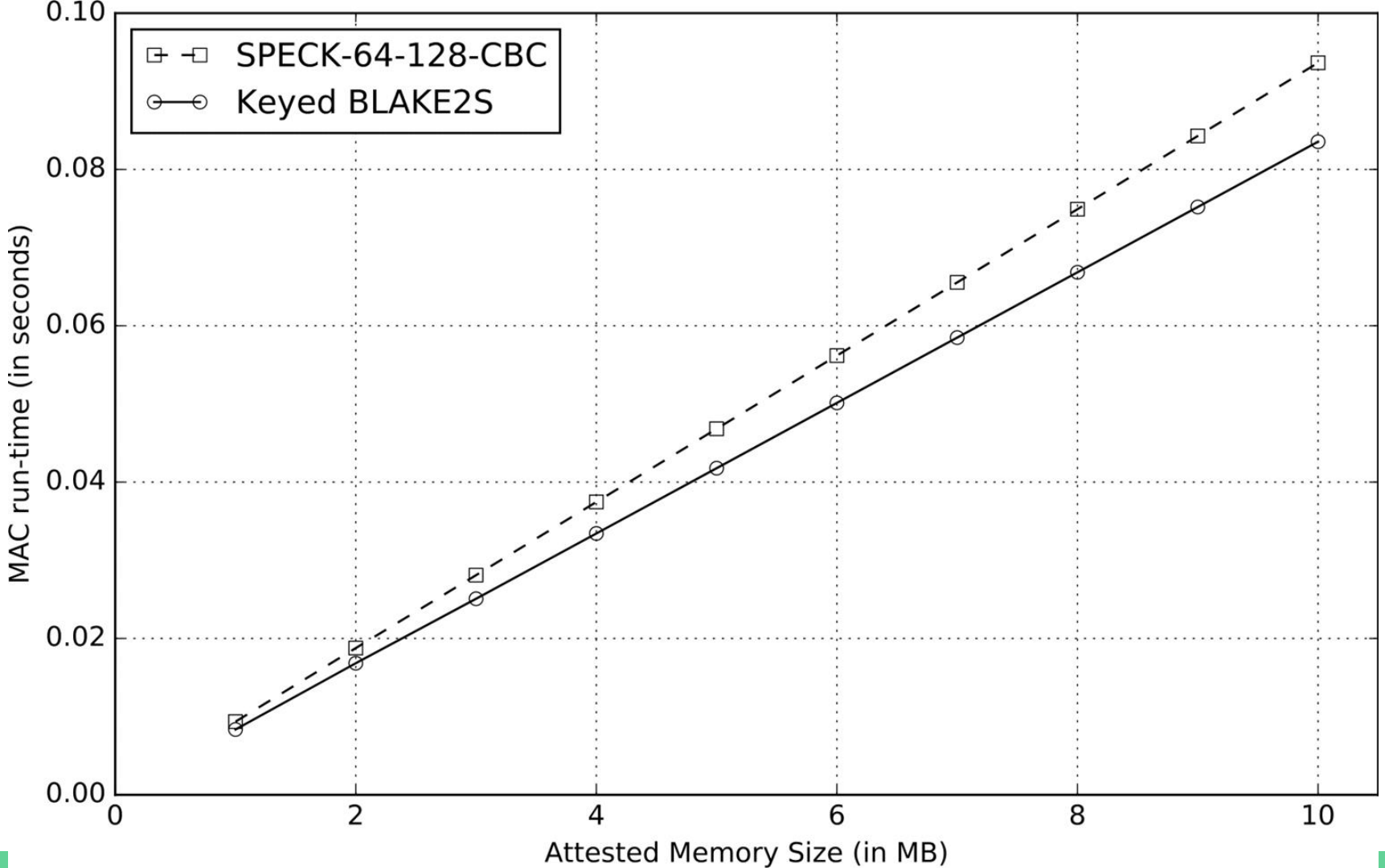
http://www.hardkernel.com/main/products/prdt_info.php

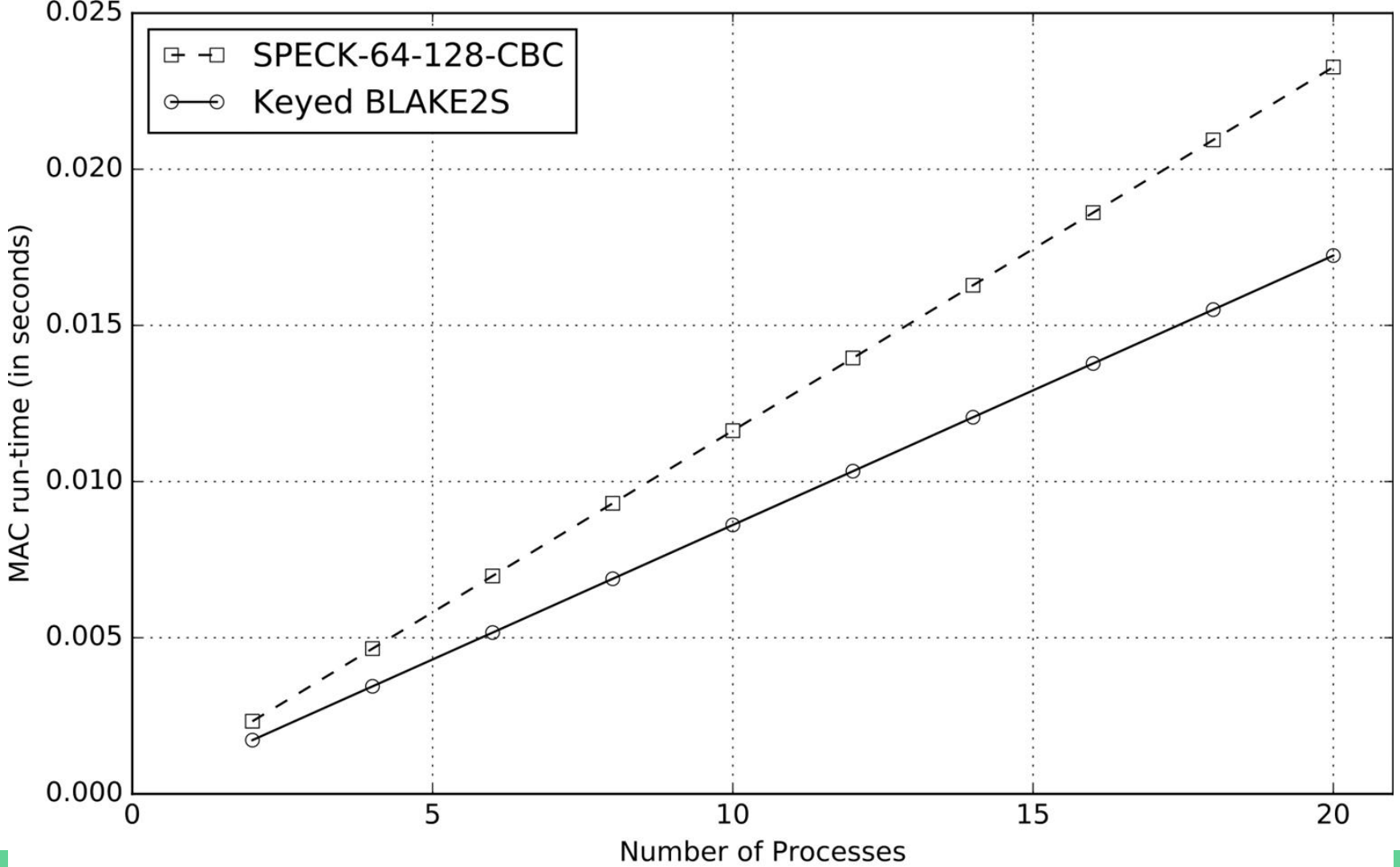
- Existing secure boot mechanism in Sabre Lite

Ensuring Freshness of Attestation Requests

- ★ No real-time clock driver implementation in Sabre Lite.
- ★ Workaround by generating pseudo-timestamp using a counter + a secondary storage
  - Upon starting, PR_{Att} loads T_0 that was saved before the last reboot.
  - When the **first** request arrives, check its timestamp (T_1) with T_0
  - Verify request. If success, keep track of T_1 and start counter.
  - $TS = T_1$ + counter value
  - Periodically store TS


HMAC-SHA-256
SIMON-64-128-CBC
AES-128-CBC
Keyed BLAKE2S
SPECK-64-128-CBC
MAC run-time (in seconds)
0.00
0.01
0.02
0.03
0.04
0.05
0
5
10
15
20
Number of Processes


SPECK-64-128-CBC
Keyed BLAKE2S
MAC run-time (in seconds)
0.00
0.02
0.04
0.06
0.08
0.10
0
2
4
6
8
10
Attested Memory Size (in MB)


SPECK-64-128-CBC
Keyed BLAKE2S
MAC run-time (in seconds)
0.025
0.020
0.015
0.010
0.005
0.000
0
5
10
15
20
Number of Processes