KU LEUVEN

Aion: Enabling Open Systems through Strong Availability Guarantees for Enclaves

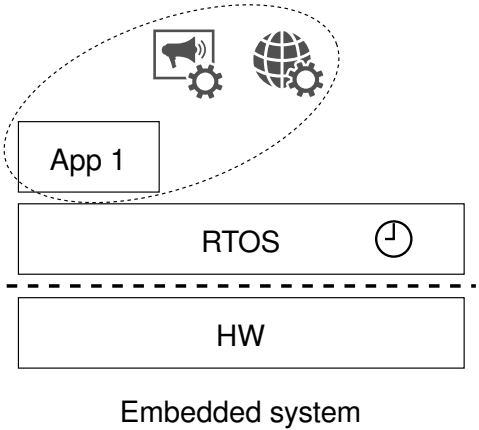
Aion technical talk

Published at CCS '21

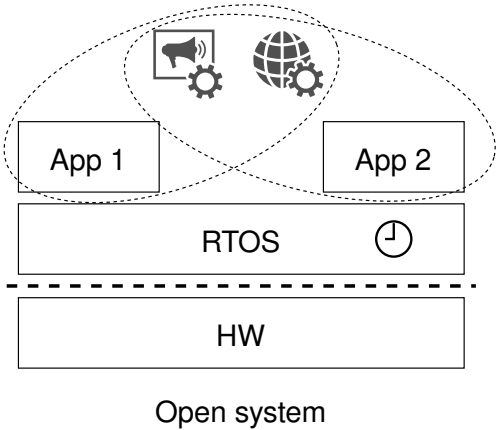
Fritz Alder, Jo Van Bulck, Jan Tobias Mühlberg, Frank Piessens
imec-DistriNet, KU Leuven

October 19, 2021

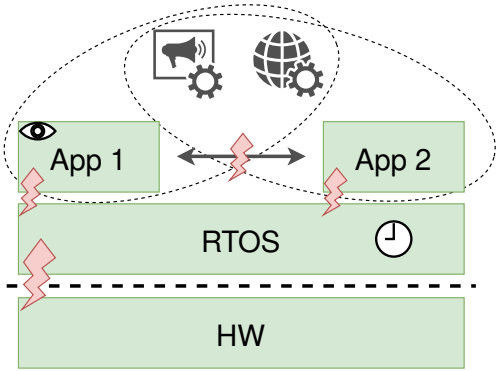
Embedded system overview



Modern and open system overview



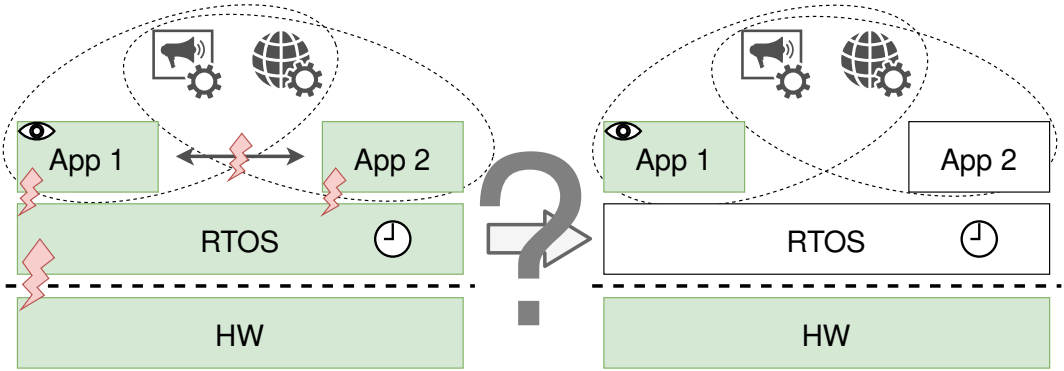
Modern and open system – Who do we have to trust?



TCB for availability
in open system

- Monopolizing a system resource or stalling the CPU is often possible.
- **Hackers do not cooperate.**
- **Even postponing deadlines** can have harsh consequences.

Modern and open system – What do we want?



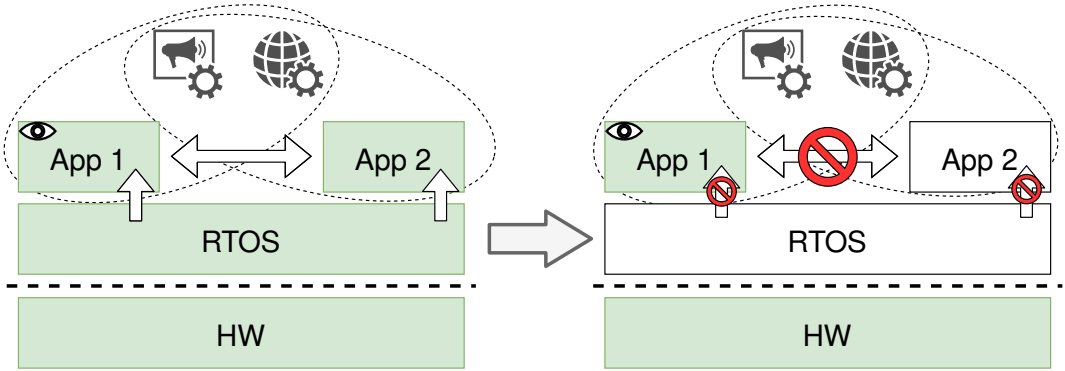
TCB for availability in open system

With TEE?

Trusted Execution Environments: A castle inside the processor

Trusted Execution Environments: Only allow strictly defined access

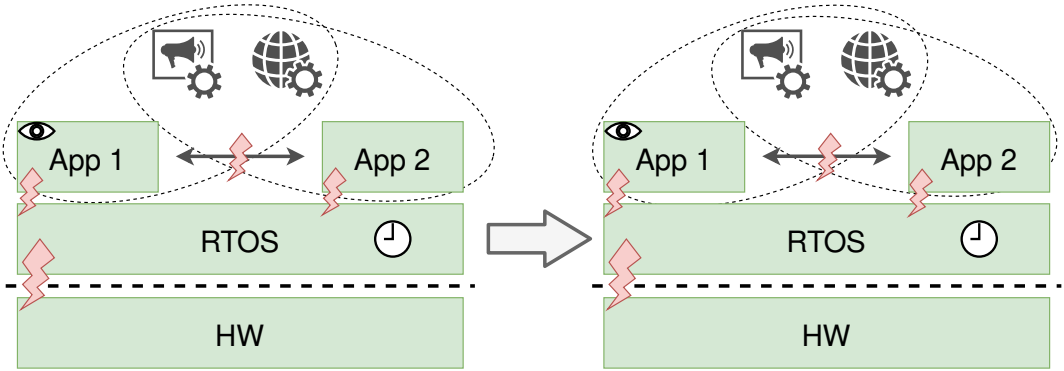
Trusted execution: Good for confidentiality and integrity



TCB for confidentiality/integrity in open system

Good spatial isolation with TEEs

Trusted execution: Not good for availability



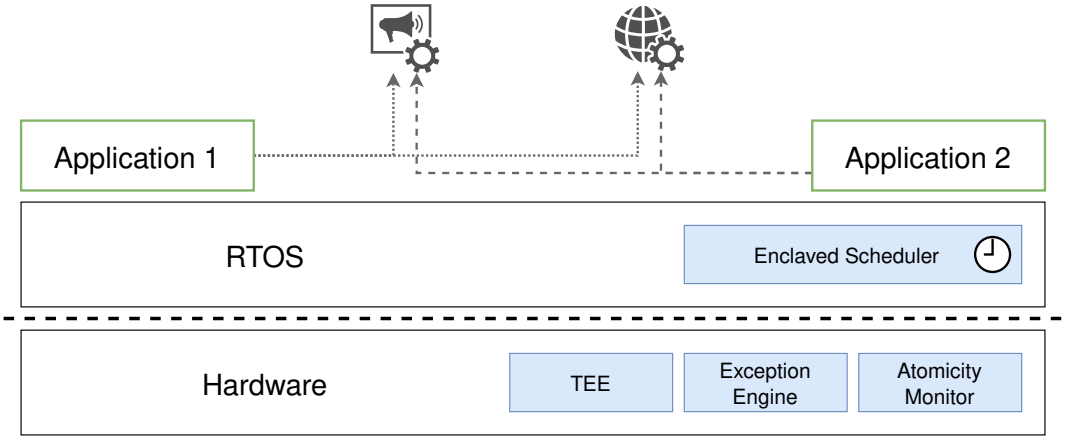
TCB for availability in open system

No temporal isolation with current TEEs

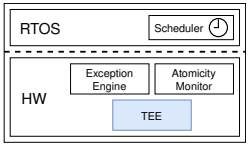
Aion Contributions in Short

- Security architecture that extends TEEs with **guarantees** on enclave availability, **even in the presence of software adversaries**.
- Progress and real-time guarantees can be offered to a number of applications of the *same* priority.
- **Decoupling** of availability guarantees from confidentiality and integrity guarantees.
- Prototype implementation with the RIOT OS and Sancus.

Aion Architecture Overview

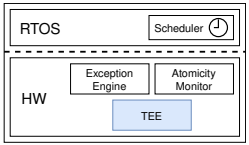


Aion Design – TEE



- Aion builds on embedded TEE architectures for **isolation**, **attestation**, and **dynamic enclave deployment**.
- Sancus is a suitable candidate (16-bit, OSS, simple architecture).

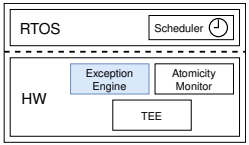
Aion Design – TEE



- Aion builds on embedded TEE architectures for **isolation**, **attestation**, and **dynamic enclave deployment**.
- Sancus is a suitable candidate (16-bit, OSS, simple architecture).

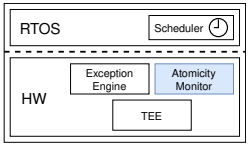
- Additional TEE features required by Aion:
  - Interruptible/restartable cryptographic operations
  - Security policy violations can not reset the system but must clear the CPU state **without side-effects**.

Aion Design – Exception Engine



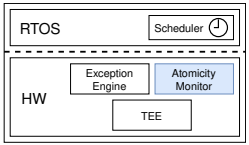
- Goal: **Securely** interrupt enclaves and pass control to the scheduler.
- Two possible types of exceptions are handled:
  - **Interrupts** of peripherals (timer, sensors, etc)
  - **Violations** of security or availability policies
- **But: Do not fully trust the scheduler**. Saving and restoring context is done by hardware and application enclaves respectively.

Aion Design – Atomicity Monitor



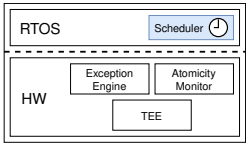
- Goal: Only the scheduler should be in full control over the availability of the platform.
- **Disabling interrupts** should not be possible.

Aion Design – Atomicity Monitor



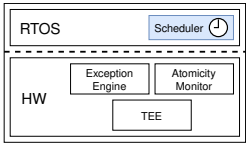
- Goal: Only the scheduler should be in full control over the availability of the platform.
- **Disabling interrupts** should not be possible...**but atomic sections are necessary!**
- Introduce instruction for bounded atomicity (`clix`).
- Nesting or exceeding allowed `clix` length results in atomicity violations.
- **Enclave entries are difficult!**

Aion Design – Scheduler



- Goal: Deterministic and timely response to events.
- Previous modules:
  - TEE: Enclaves ensure spatial isolation of applications.
  - Exception Engine: All events reach their handler without security compromise.
  - Atomicity Monitor: Bound atomicity to limit latency of event $\rightarrow$ event handler.

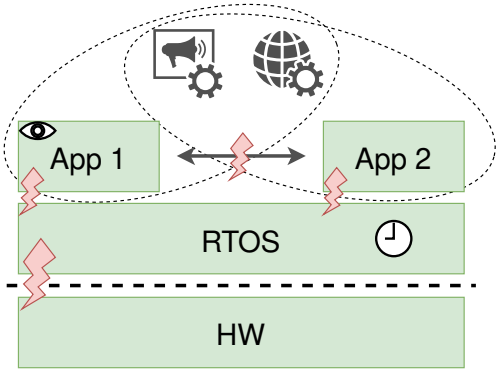
Aion Design – Scheduler



- Goal: Deterministic and timely response to events.
- Previous modules:
  - TEE: Enclaves ensure spatial isolation of applications.
  - Exception Engine: All events reach their handler without security compromise.
  - Atomicity Monitor: Bound atomicity to limit latency of event → event handler.
- **Enclaved Scheduler** is registered as handler **for all interrupt types**.
- **But: Scheduler is not trusted for confidentiality**
  → Must not have access to peripheral data or MMIO data region.

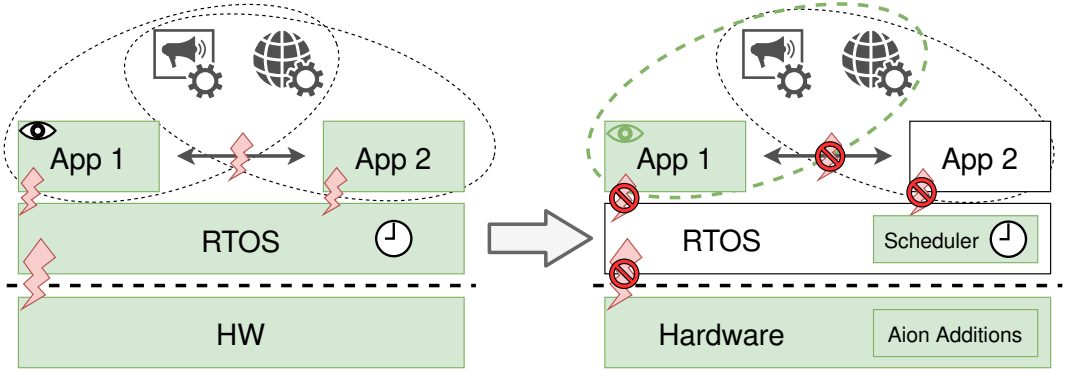
Result: Scheduler handles all events after a bounded time delay.
Guaranteed by hardware.

TCB for Availability – Before Aion



TCB for availability in TEEs without Aion

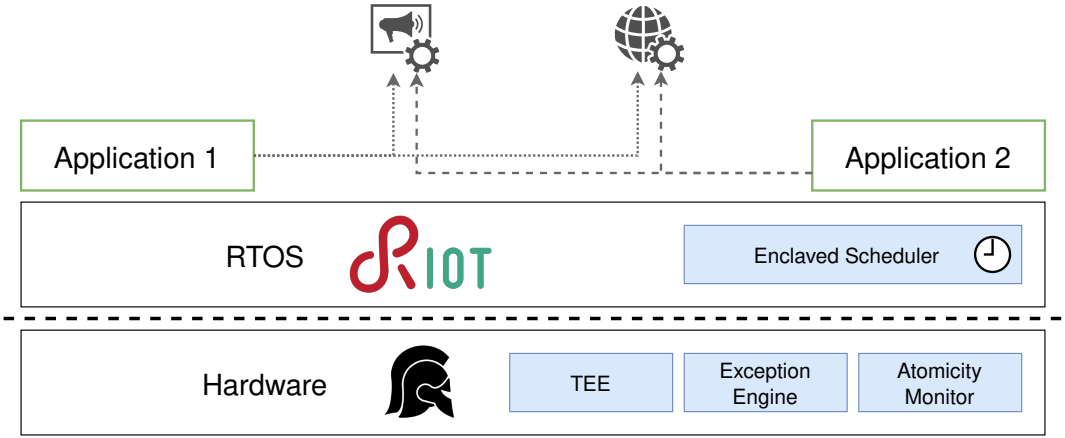
TCB for Availability – With Aion



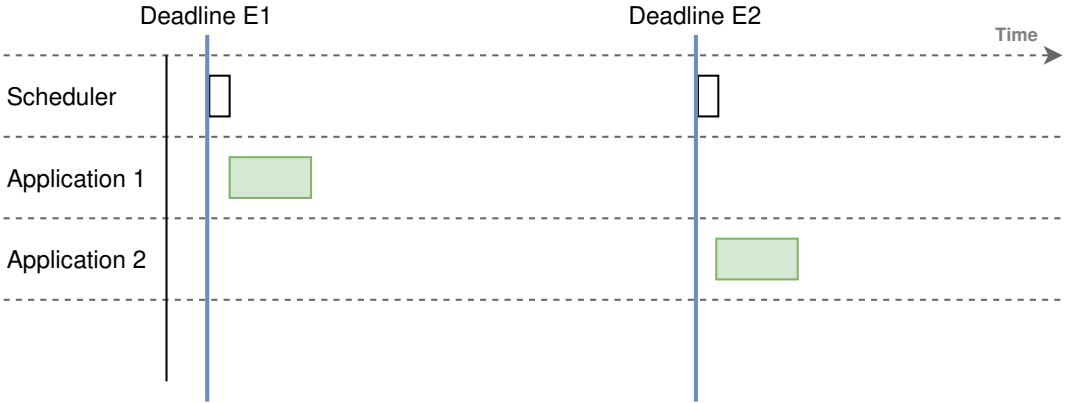
TCB for availability in TEEs without Aion

TCB for availability with Aion

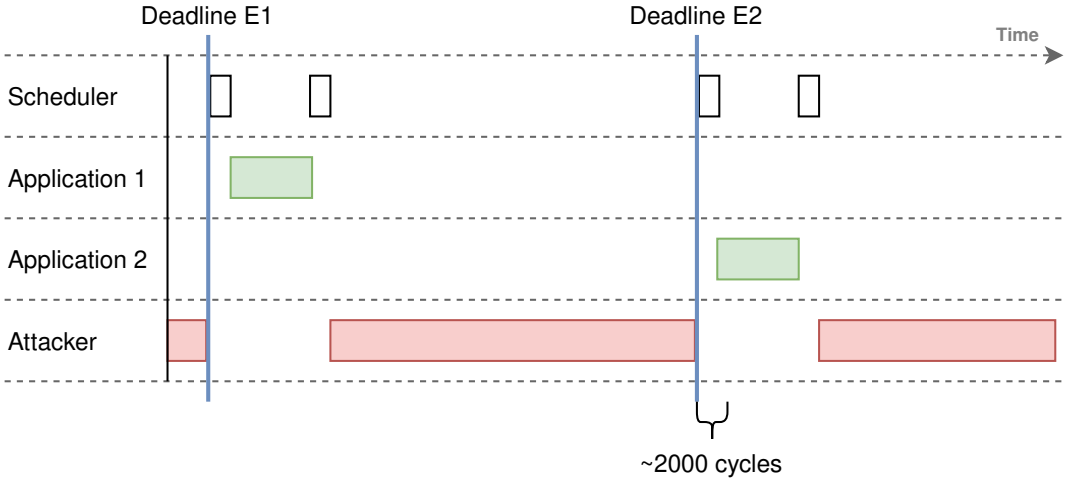
Aion Prototype Implementation



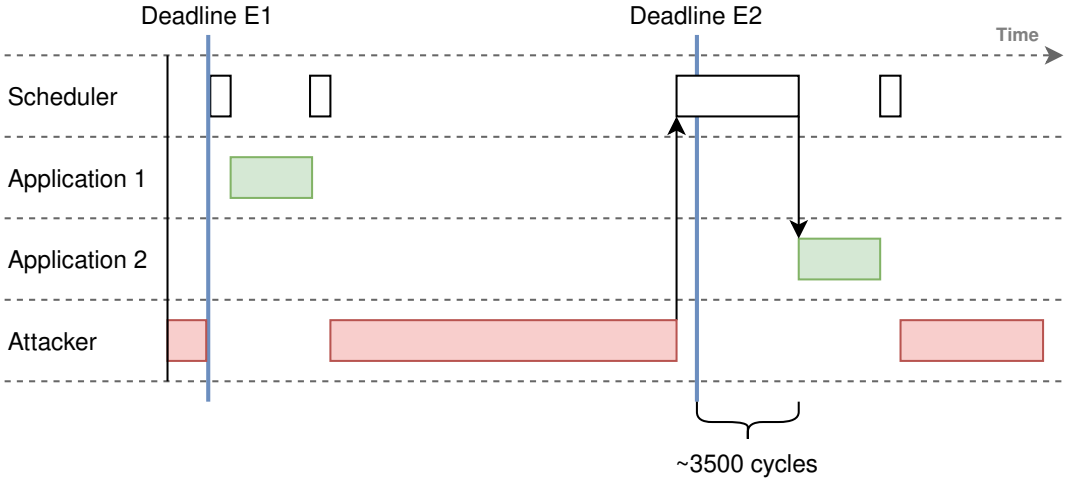
Aion Results – Case study with activation deadlines



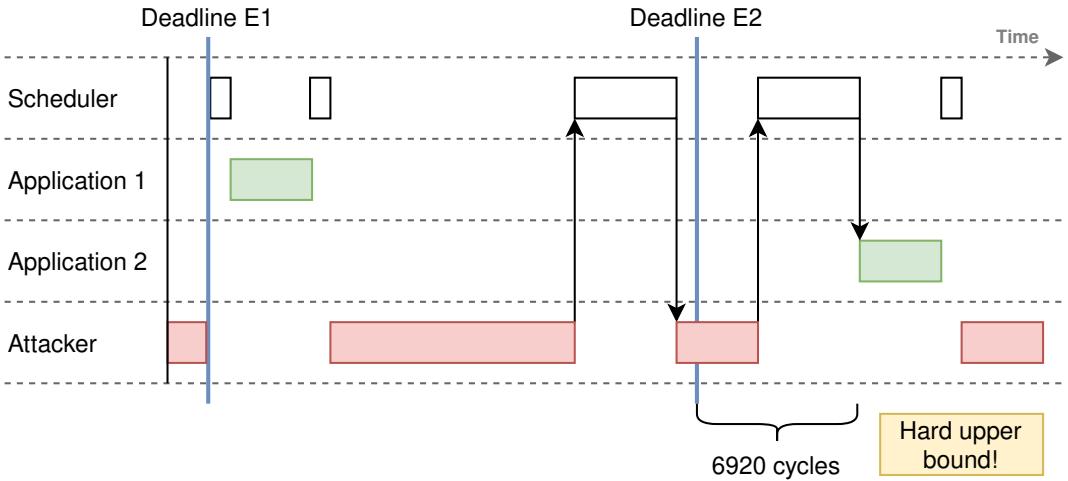
Aion Results – Case study with activation deadlines



Aion Results – Activation deadlines under attack



Aion Results – Activation deadlines worst case attack



Aion Limitations

- Aion only guarantees an **Interrupt Arrival Time** of 6920 cycles
- After this, the handling job starts to execute with its own atomically bounded periods
  $\rightarrow$ **Guaranteeing progress is not trivial!**

Aion Limitations

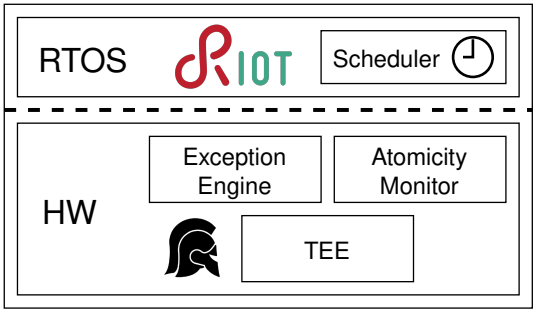
- Aion only guarantees an **Interrupt Arrival Time** of 6920 cycles
- After this, the handling job starts to execute with its own atomically bounded periods
  $\rightarrow$ **Guaranteeing progress is not trivial!**
- Right now: Progress = `clix` bound (1000 cycles)
- Future work: Let scheduler mask interrupts/disable interrupts for high-priority events

Aion Limitations

- Aion only guarantees an **Interrupt Arrival Time** of 6920 cycles
- After this, the handling job starts to execute with its own atomically bounded periods
  $\rightarrow$ **Guaranteeing progress is not trivial!**
- Right now: Progress = `clix` bound (1000 cycles)
- Future work: Let scheduler mask interrupts/disable interrupts for high-priority events
- Additionally: **Restartable crypto needed!** Interrupting crypto operations that exceed `clix` length have no chance in the presence of adversaries

Conclusion

- Extend TEE architectures with an **Exception Engine** and an **Atomicity Monitor** to enable a **Protected Scheduler**.
- Our implementation provides **deterministic scheduling** and **trusted time**, **even in the presence of strong software adversaries**.



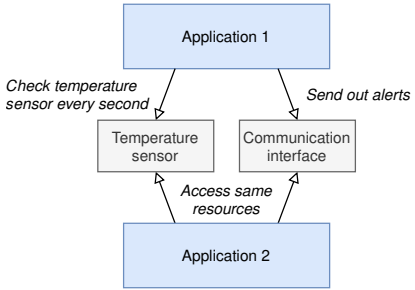
Aion can guarantee interrupt arrival latencies of 6920 cycles (346ns @ 20Mhz).

Weblinks

- Link to paper: https://falder.org/files/paper/2021_aion.pdf
- GitHub repository with all code, case study, and hardware changes: https://github.com/sancus-tee/sancus-riot

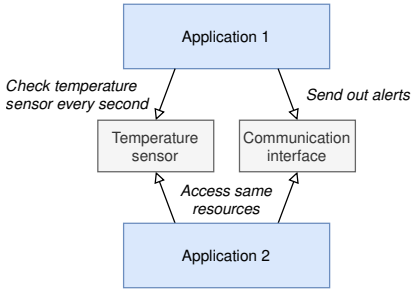
Backup slides

Open system – Security and availability requirements



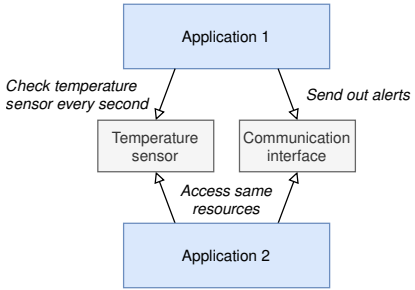
Open system – Security and availability requirements

- Bounded activation latency
- Guaranteed progress



Open system – Security and availability requirements

- Bounded activation latency
- Guaranteed progress
- Guaranteed device access
- Safety independence
- No trust hierarchy



	Masti	TrustLite	TyTAN	Sancus
Bounded activation latency	✔	–	–	–
Guaranteed progress	◐	◐	◐	◐
Guaranteed device access	✔	◐	◐	◐
Safety independence	–	–	–	–
No trust hierarchy	–	–	–	–
Architecture	AVR	Siskiyou Peak		MSP430

	Masti	TrustLite	TyTAN	Sancus	Aion
Bounded activation latency	✔	–	–	–	✔
Guaranteed progress	◐	◐	◐	◐	✔
Guaranteed device access	✔	◐	◐	◐	✔
Safety independence	–	–	–	–	✔
No trust hierarchy	–	–	–	–	✔
Architecture	AVR	Siskiyou Peak		MSP430	

Aion Results – Scheduler operations

Scheduler operation	Best case (cycles)	Worst case (cycles)
Create job	688	860
Exit job	512	736
Sleep	1124	**1320**
Yield	424	628
Get time	212	

Table: Detailed overhead in cycles for the operations provided by the scheduler.

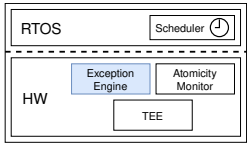
Aion Results – Scheduler delay

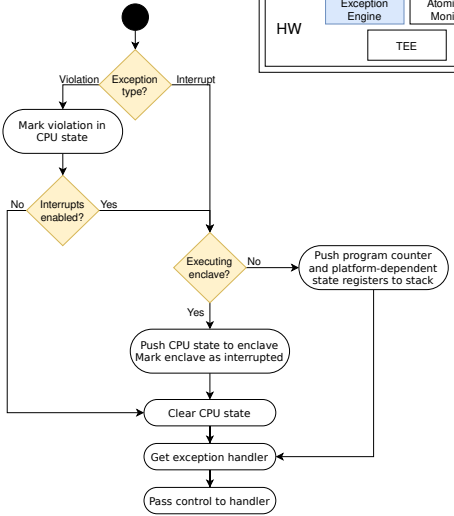
Task/Stage	Best case (cycles)	Worst case (cycles)
1. Interrupt arrival	0	10 + clix + **1320**
2. Interrupt processing	7	(35)
3. Scheduler entry	157	(115)
4.1 Timer	1356	4075
4.2 Scheduler run	443	443
5 Scheduler resume	72	72
Activation latency	2035	5920 + clix

Table: Detailed overhead in cycles for an event that preempts a running job. Shown are measurements with default Aion parameters and the overheads in the best and worst case.

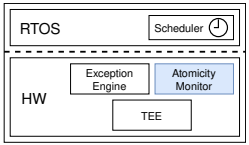
Aion Design – Exception Engine

- Distinguish between exception types
- Distinguish between protection modes:
  - Unprotected: Default bahvior
  - Enclave: Push to TCS
- Put violation marker in CPU state if enclave is resumed later
- Do not store violations in TCS if interrupts are disabled

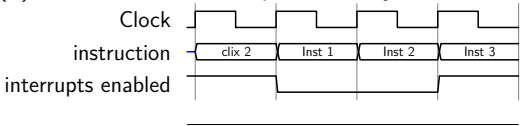




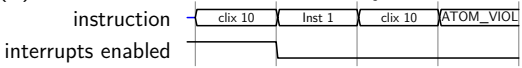
Aion Design – Atomicity Monitor



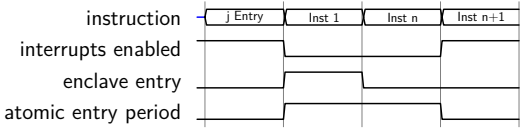
(a) clix disables interrupts for x cycles



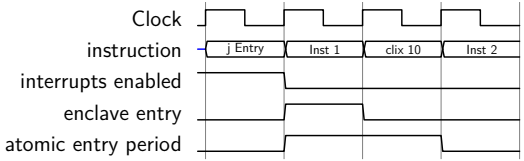
(b) Nested clix result in atomicity violation



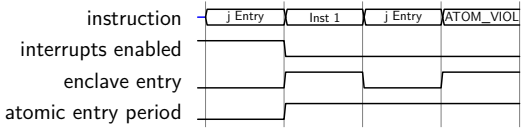
(c) Interrupts are disabled on enclave entries

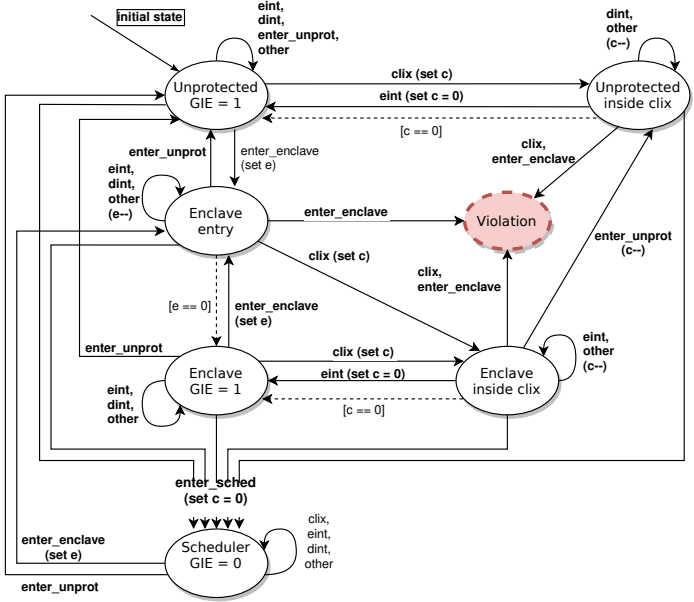


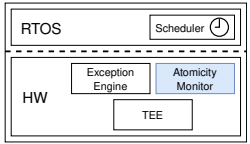
(d) Use clix after enclave entries



(e) Prolonging atomic entries gives violation





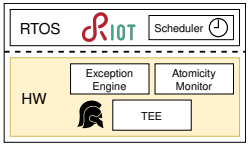


State transitions:
- clix
- eint
- dint
- enter_unprot
- enter_sched
- enter_enclave
- other

Variables:
e = Counter for atomic enclave entry
c = Counter for clix
Legend:
⟶ a (b)
Transition on a with side effect b
- - - -➤ [a]
Automatic transition on condition a

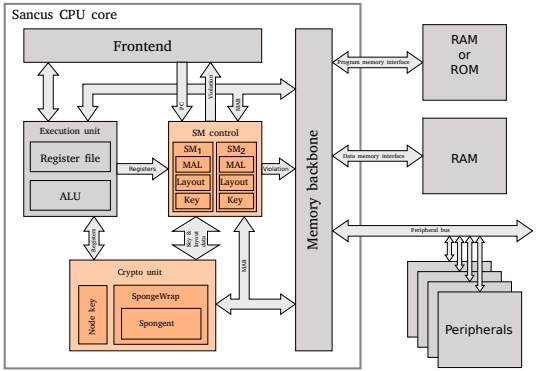
Atomicity States

Sancus Background



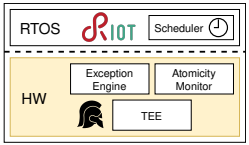
Extends openMSP430 with strong security primitives

- Software Component Isolation
- Cryptography & Attestation
- Secure I/O through isolation of MMIO ranges
- Efficient, Modular, $\leq$ 2 kLUTs
- Cryptographic key hierarchy for software attestation
- Isolated components are typically very small ($<$ 1kLOC)



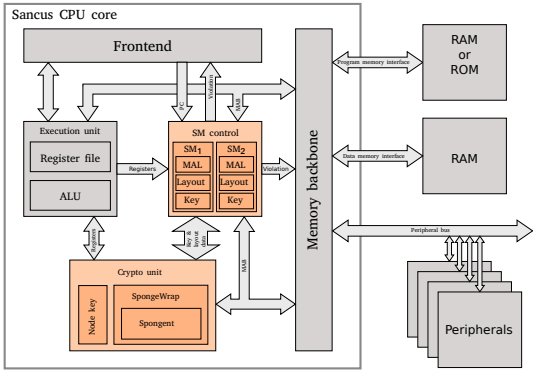
Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

Aion Prototype – Sancus

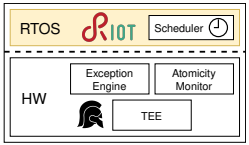


Sancus Modifications:

- Preemptive enclaves
- Interruptible crypto (restartable)
- Threading via software TCS
- Violations abort and clear state
- Adding two flags to status register:
  1. 'IRQ interrupted enclave' flag
  2. 'Violation marker' as per exception engine design
- SM ID 1 is privileged for CPUOFF and other relevant flags



Aion Prototype – RIOT



- Open-source OS for the IoT, running on 16-bit Sancus
- Supports real-time applications on resource-constrained devices
- Tickless, cooperative O(1) scheduling
- Highly modular, based on FreeRTOS

Aion modifications:

- Scheduler **and timer peripheral** protected in enclave
- Scheduler is non-interruptible
- Dynamic scheduling, e.g., only attested enclaves can get highest priority
- Fixed amount of priorities, threads and set timers