KU LEUVEN

# Aion: Enabling Open Systems through Strong Availability Guarantees for Enclaves

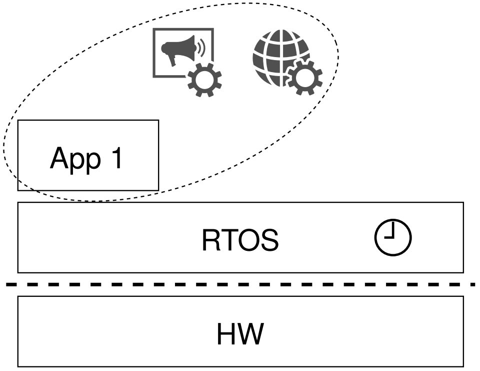
Online

CCS ’21

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

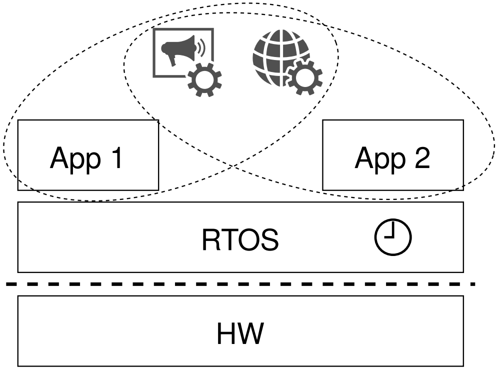
November 17, 2021

## Embedded system overview



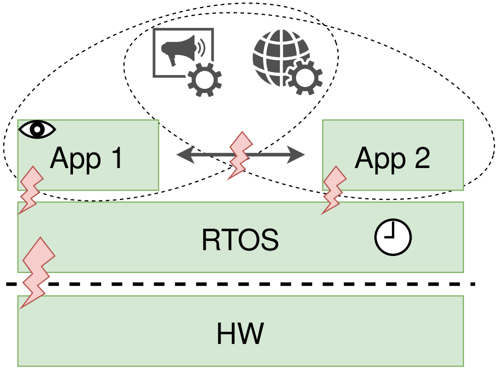
Embedded system

## Modern and open system overview



Open system

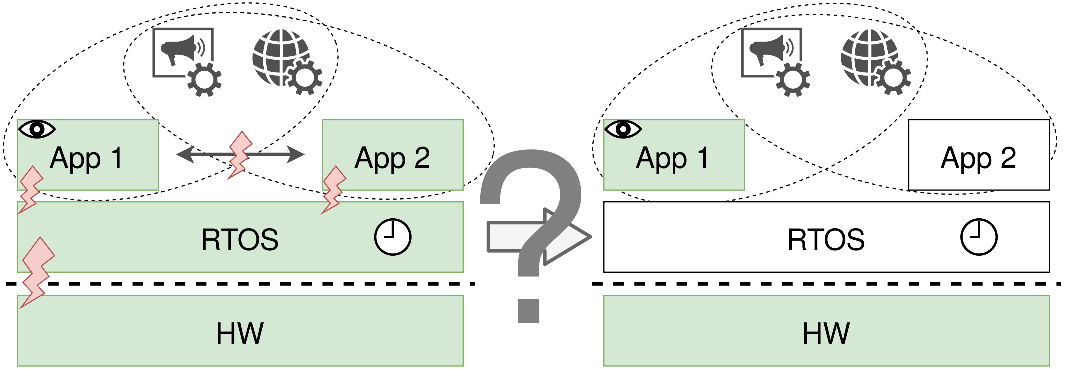
## Modern and open system – Who do we have to trust?



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?



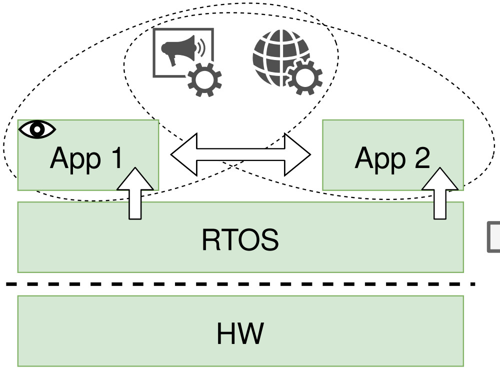
TCB for availability
in open system

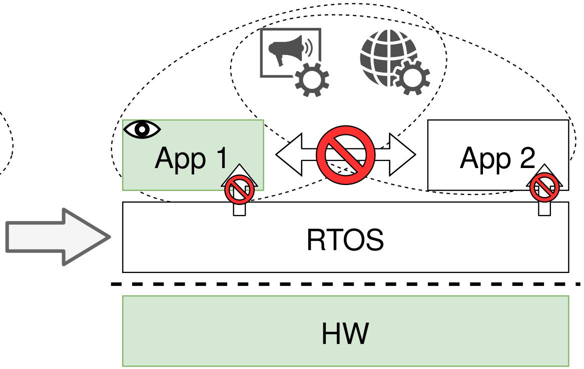
With TEE?

## Trusted Execution Environments: A castle inside the processor

## Trusted Execution Environments: Only allow strictly defined access

## Trusted execution: Good for confidentiality and integrity

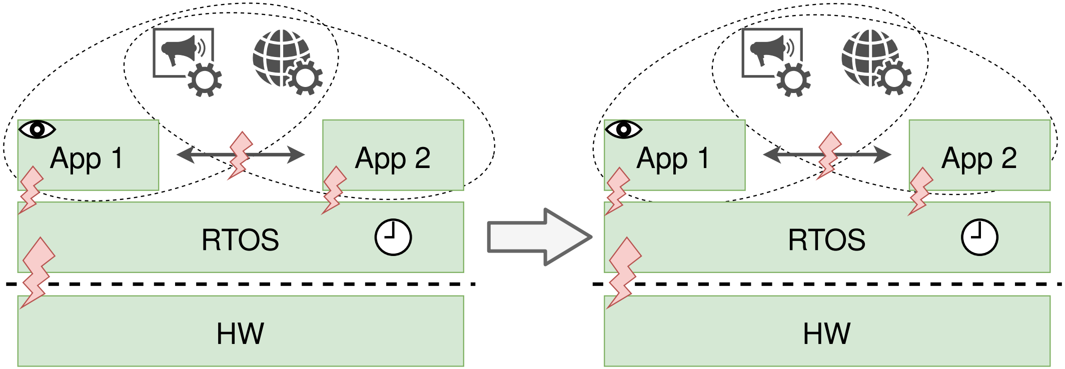




TCB for confidentiality/integrity
in open system

Good spatial isolation
with TEEs

## Trusted execution: Not good for availability



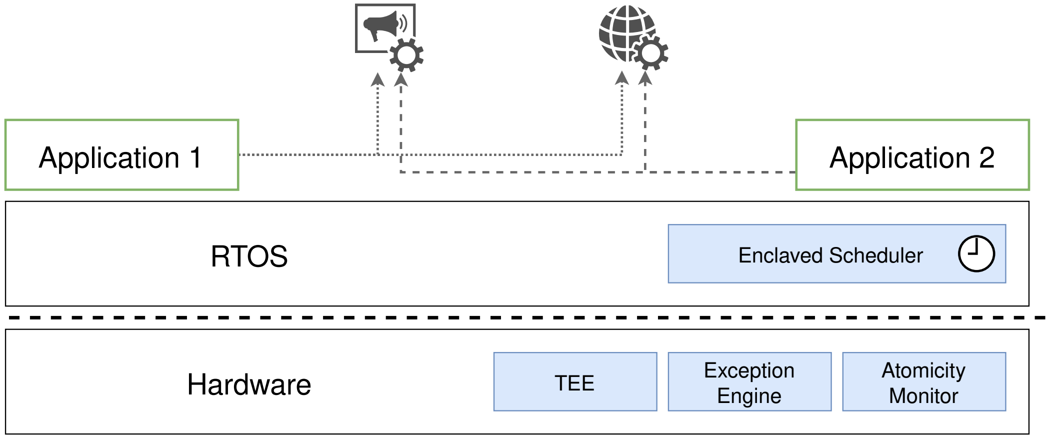
TCB for availability
in open system

No temporal isolation
with current TEEs

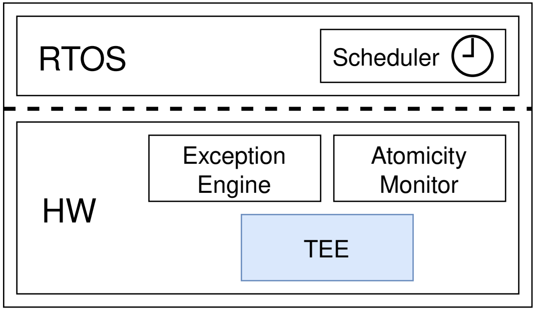
## Aion Contributions in Short

- 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

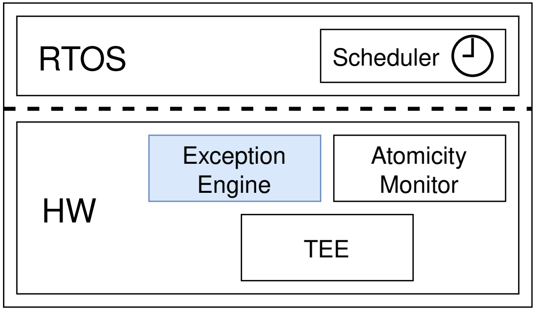


## Aion Design – TEE



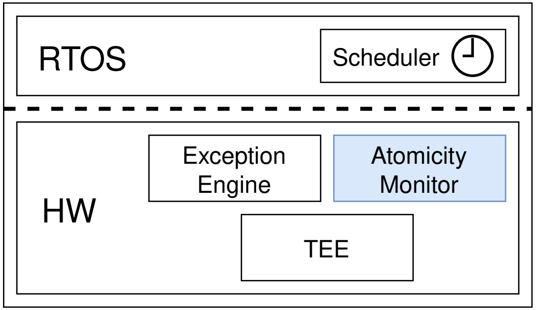
- Build on existing TEEs for:
  - isolation
  - attestation
  - dynamic enclave deployment
- TEE **violations** should not reset the system!

# Aion Design – Exception handling



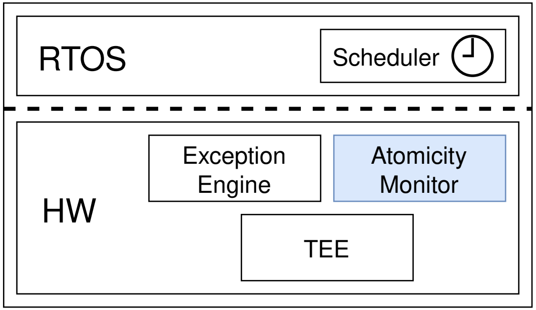
- **Securely** interrupt enclaves
- **Do not trust software handlers** with enclave data
- Instead, perform context save in hardware and context restore in Software
- Similar procedure for **violations**!
- Afterwards, jump to strictly defined handlers.

## Aion Design – Atomicity



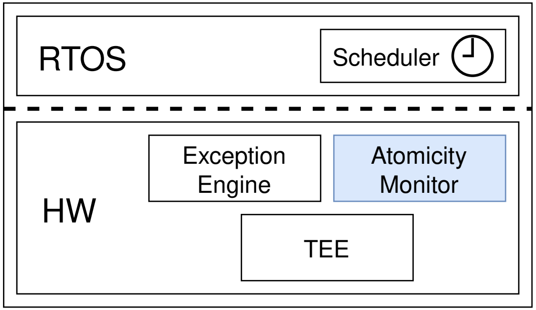
- **Disallow atomicity**

## Aion Design – Atomicity



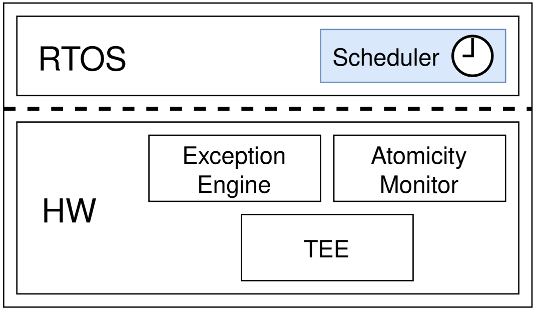
- **Disallow atomicity**
- ...except for **bounded time periods**

# Aion Design – Atomicity



- **Disallow atomicity**
- ...except for **bounded time periods**
- `clix` instruction allows to disable interrupts for $x$ cycles
- After that, interrupts will be triggered again!

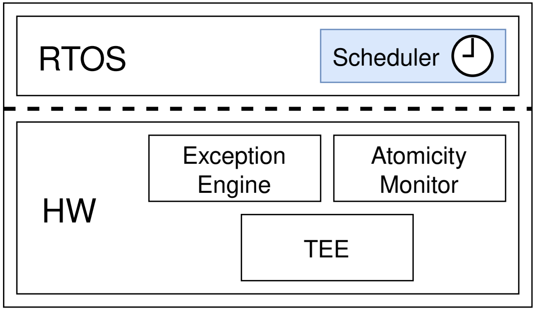
## Aion Design



- Protect scheduler via an enclave
- Ensure all interrupts are handled by the Scheduler
- Allow only the scheduler to disable interrupts completely without `clix`

$\rightarrow$ Scheduler is in complete control over platform availability

## Aion Design

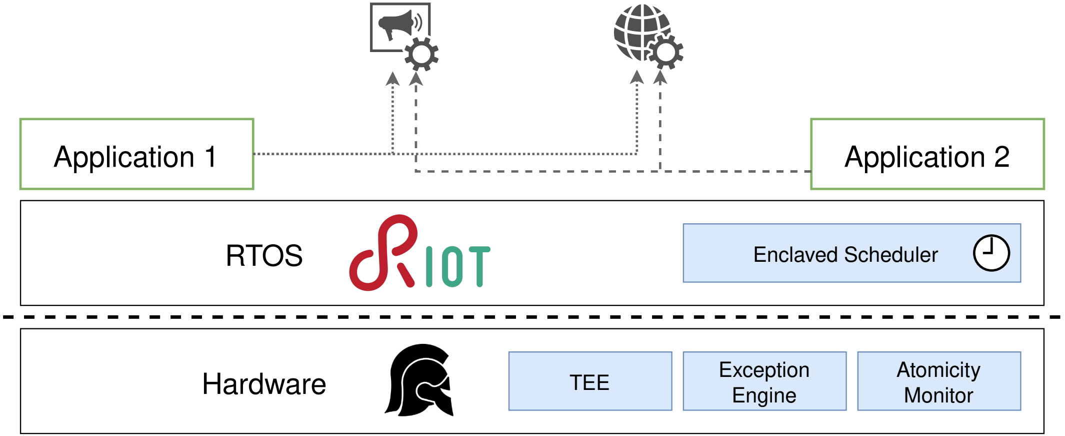


- Protect scheduler via an enclave
- Ensure all interrupts are handled by the Scheduler
- Allow only the scheduler to disable interrupts completely without `clix`

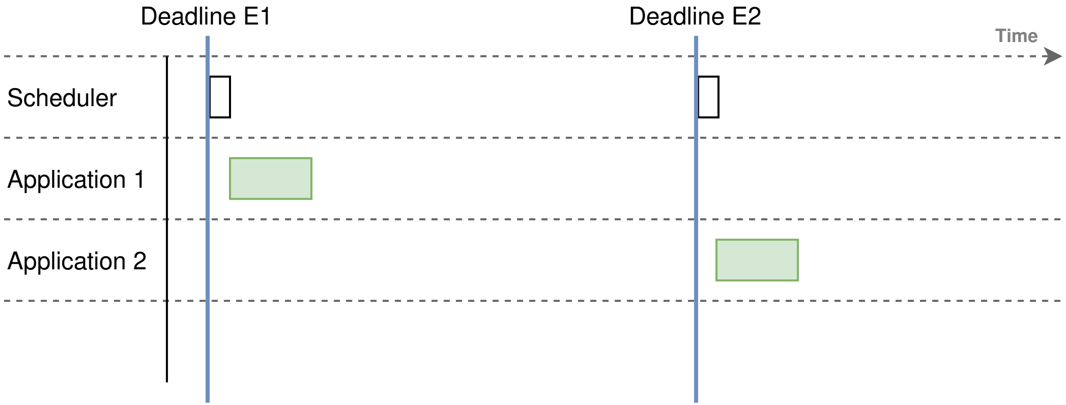
$\rightarrow$ Scheduler is in complete control over platform availability

- Scheduler can perform **guaranteed scheduling** for dynamic enclaves even **in presence of software adversaries**.

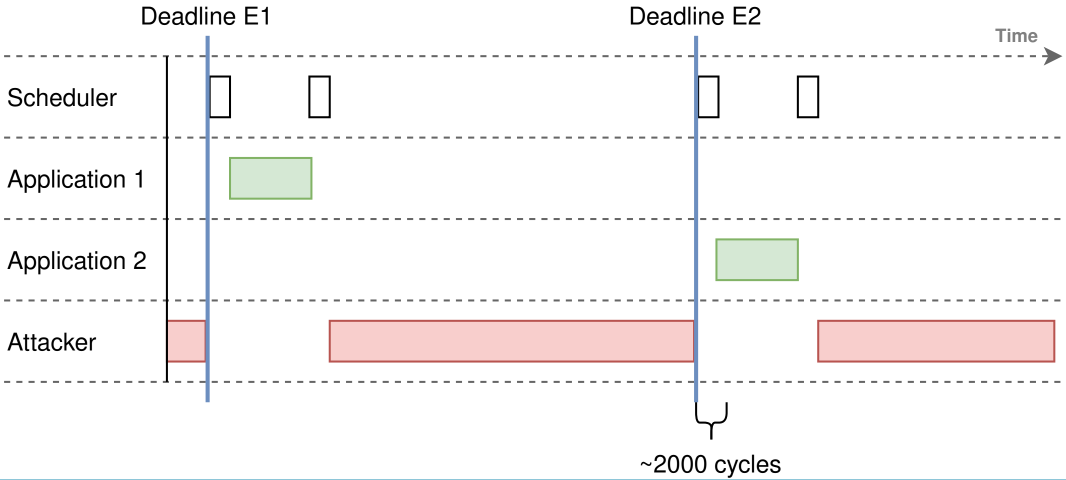
## Aion Prototype Implementation



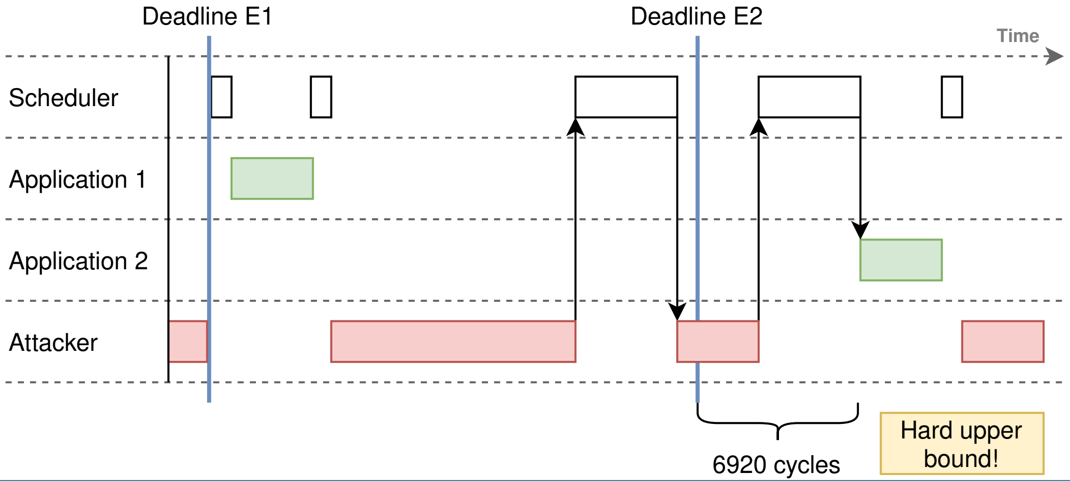
## Aion Results – Case study with activation deadlines



## Aion Results – Activation deadlines under attack



## Aion Results – Activation deadlines worst case attack

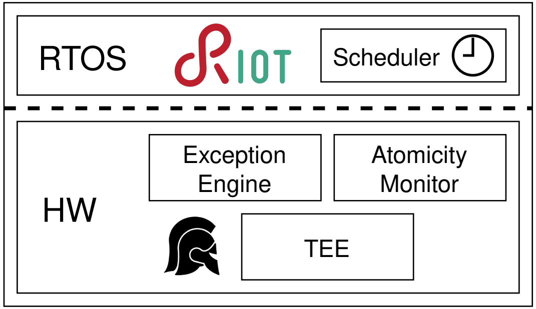


## Aion Limitations

- 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!**
- Right now: Progress = `clix` bound (1000 cycles)

## 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).
Full source code online here: https://github.com/sancus-tee/sancus-riot

KU LEUVEN

# Aion: Enabling Open Systems through Strong Availability Guarantees for Enclaves

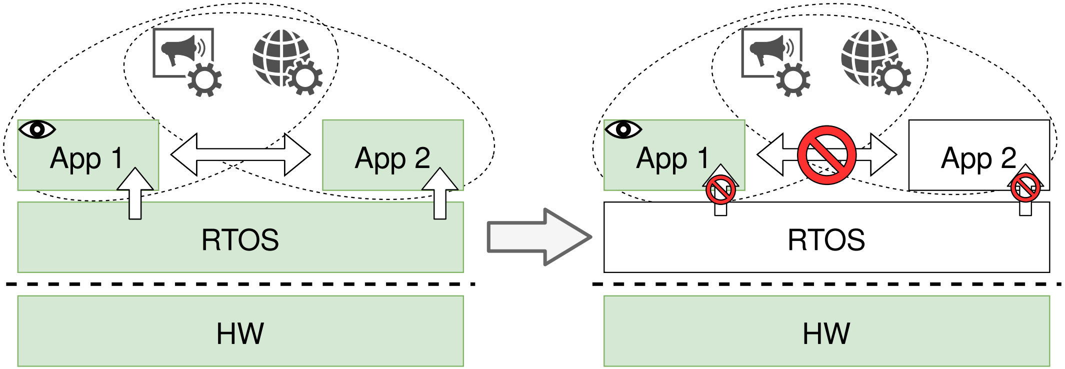
Online

CCS ’21

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

November 17, 2021

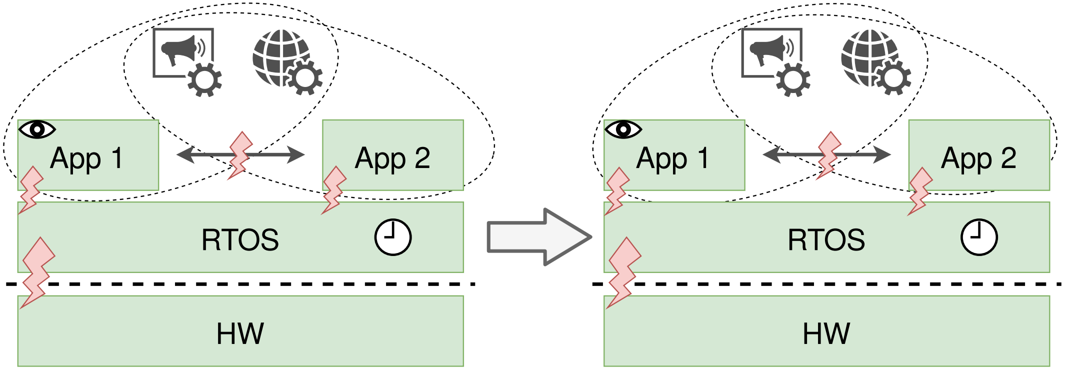
## Trusted execution: Good for confidentiality and integrity



TCB for confidentiality/integrity
in open system

Good spatial isolation
with TEEs

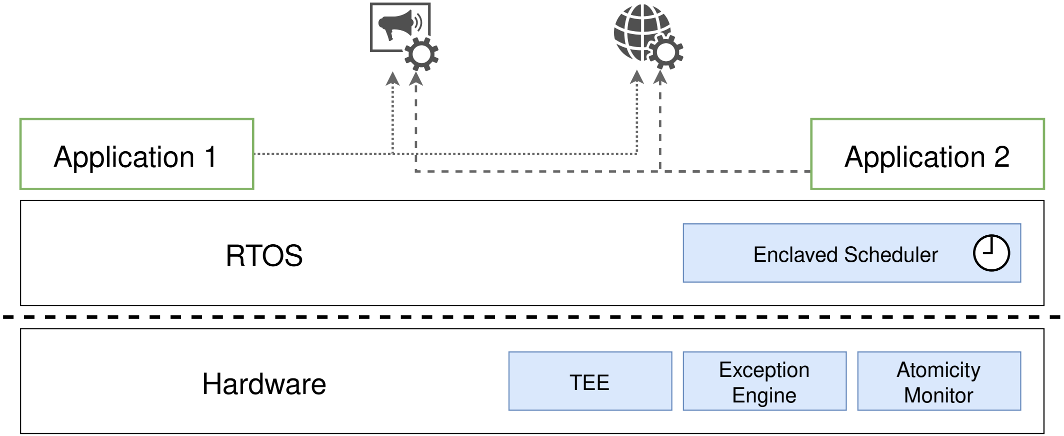
## Trusted execution: Not good for availability



TCB for availability
in open system

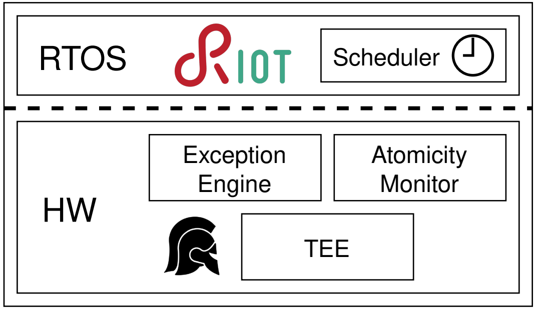
No temporal isolation
with current TEEs

## Aion Architecture Overview



## Aion summary

- 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).
Full source code online here: https://github.com/sancus-tee/sancus-riot