



School of Computer Science & Engineering  
**Trustworthy Systems Group**

# Verification Status of Time Protection and Microkit-based OS Services

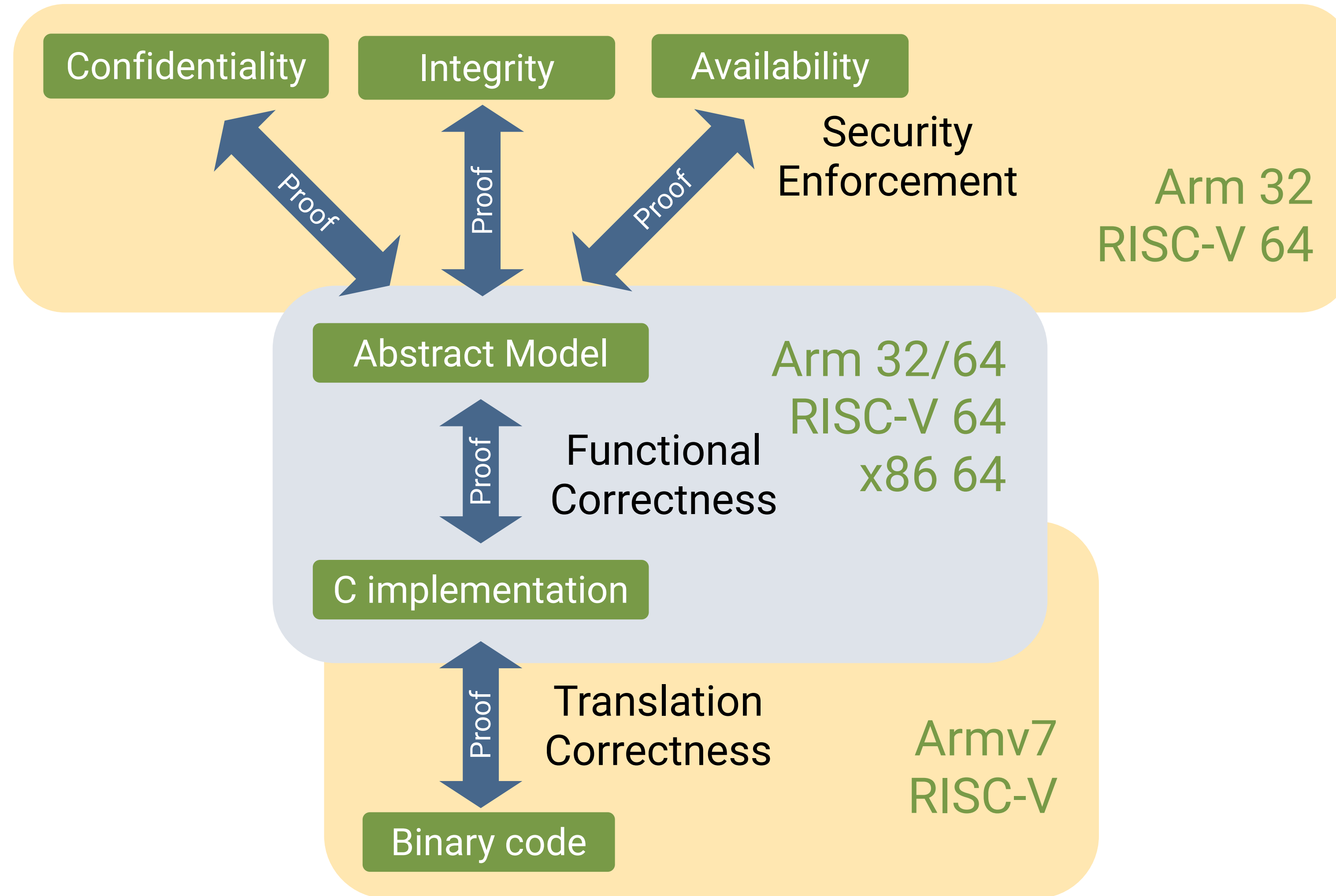
**Dr Rob Sison**

Senior Research Associate, UNSW Sydney  
[r.sison@unsw.edu.au](mailto:r.sison@unsw.edu.au)



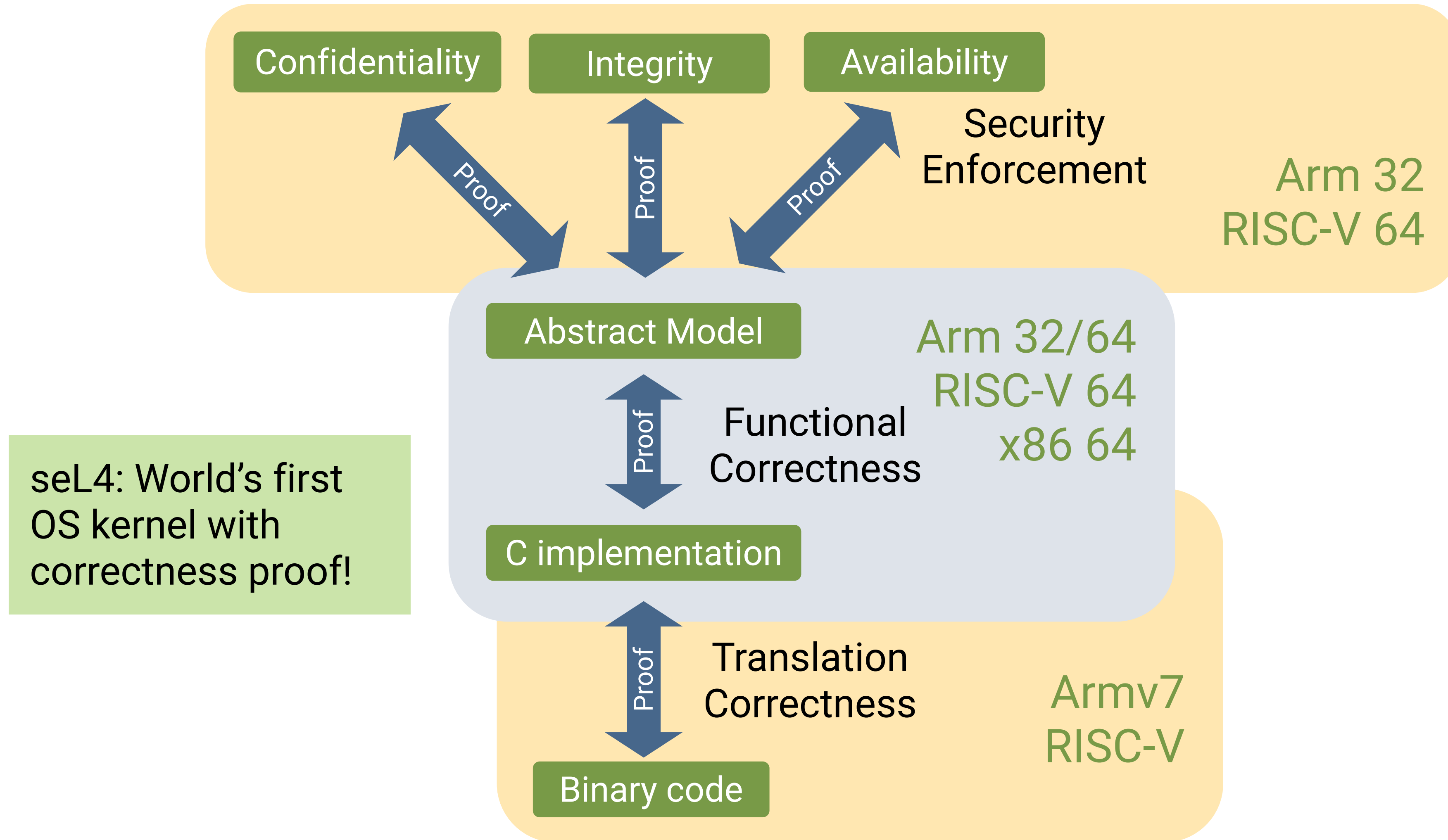


# The verified seL4 OS microkernel



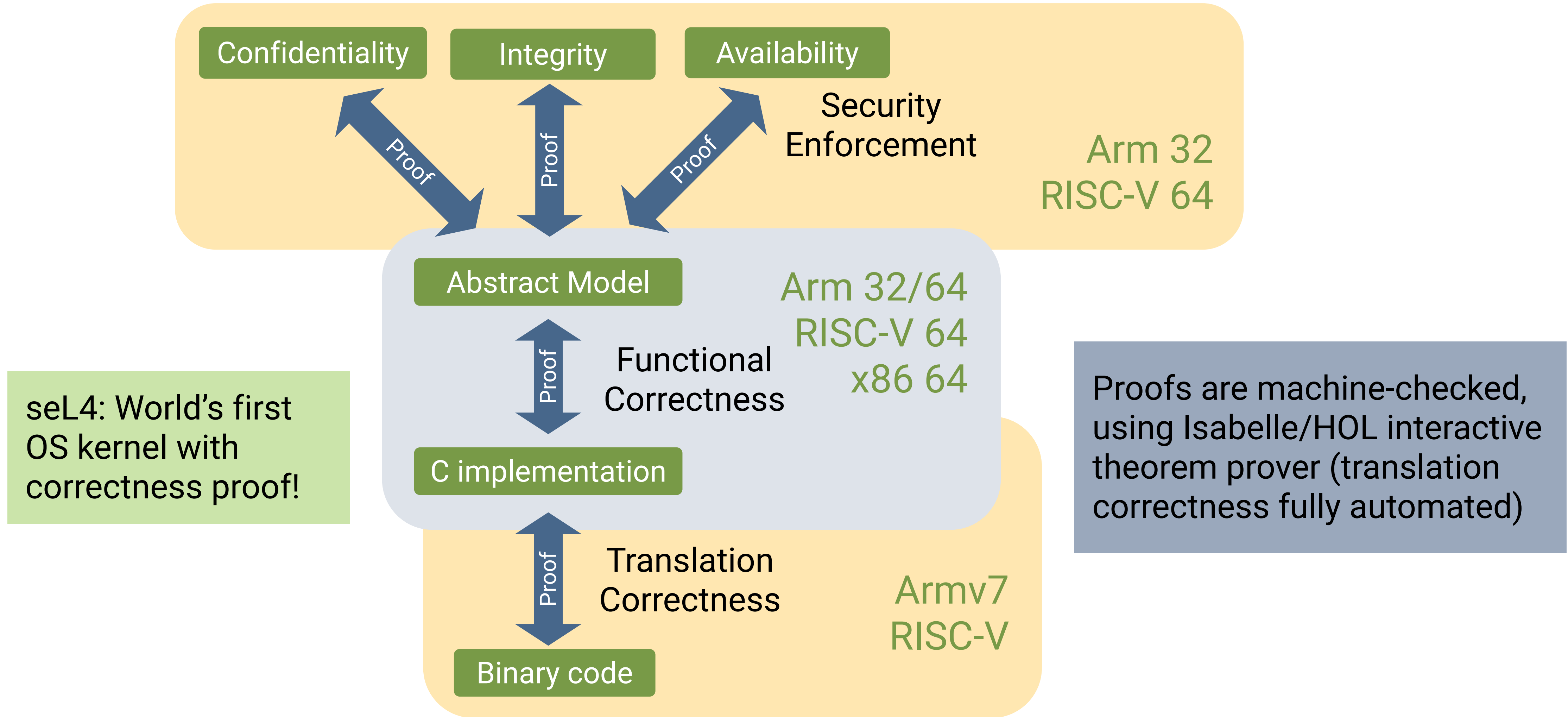


# The verified seL4 OS microkernel

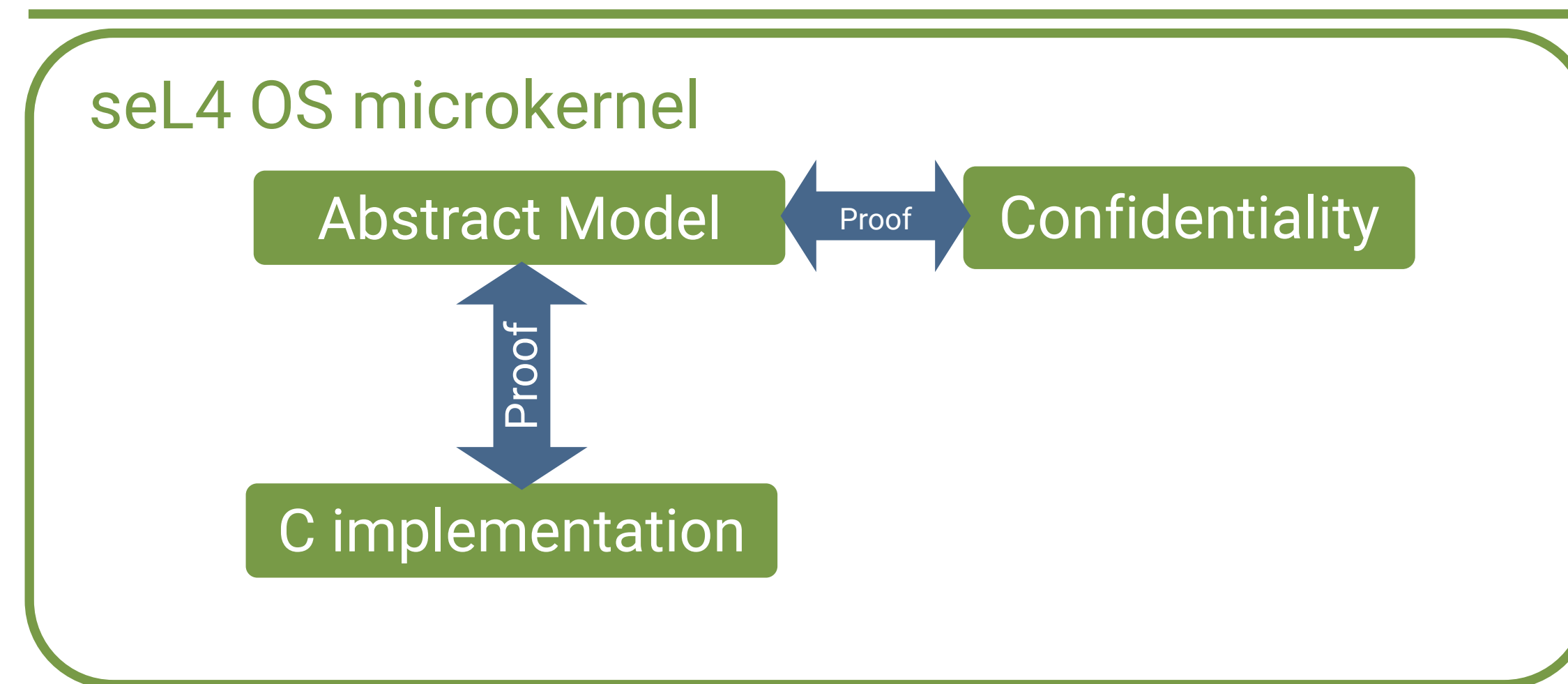




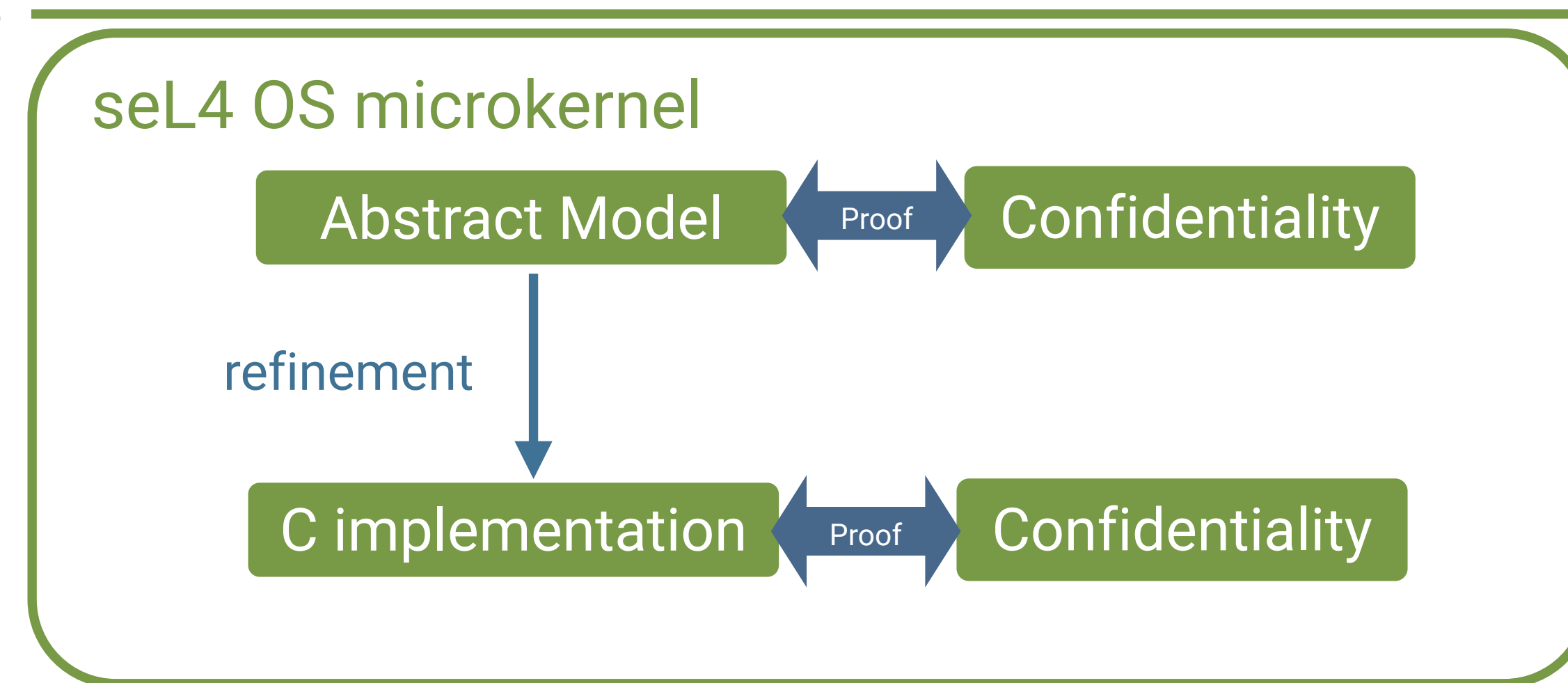
# The verified seL4 OS microkernel

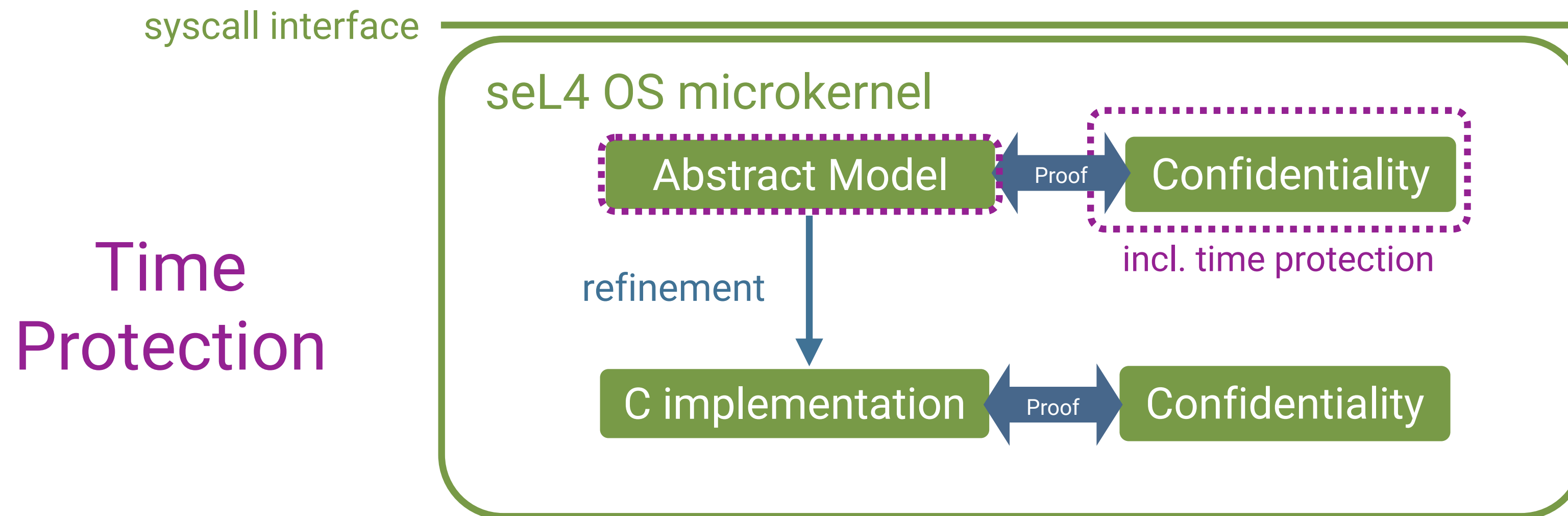


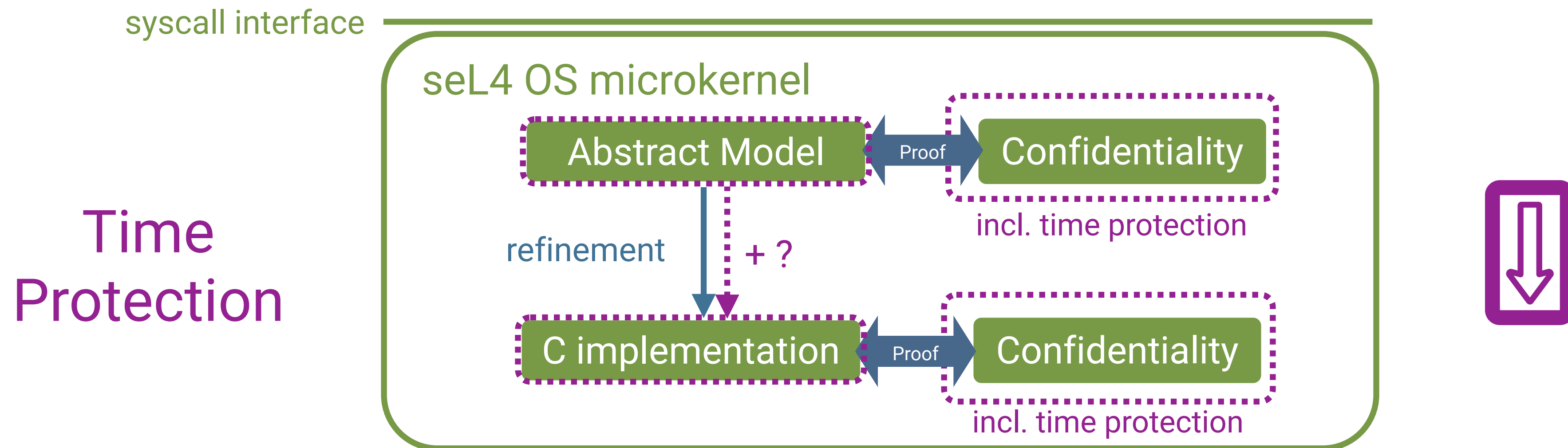
syscall interface



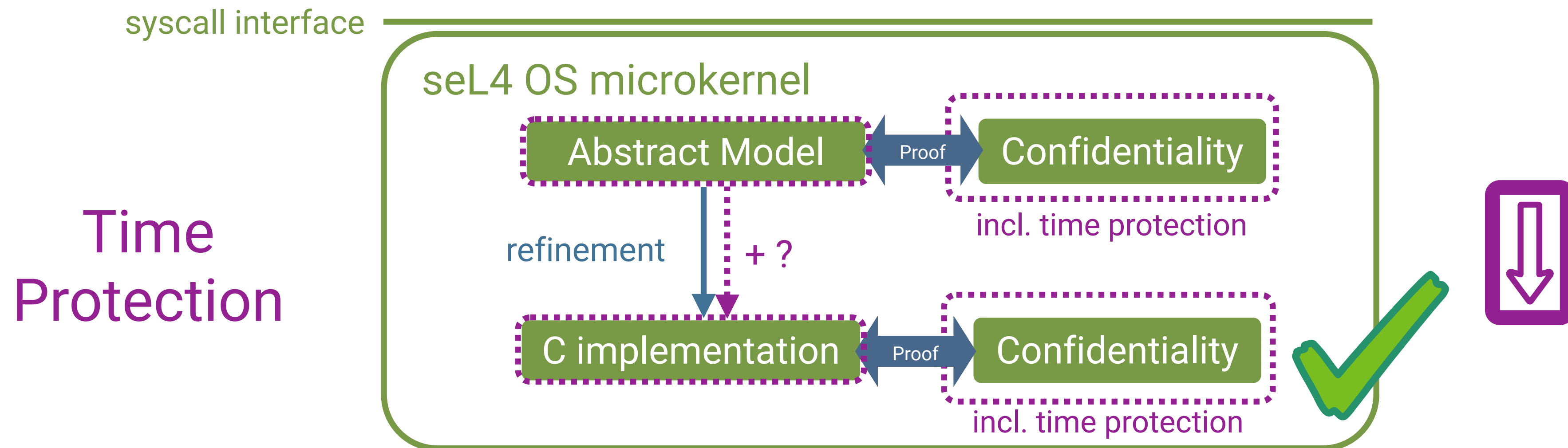
syscall interface





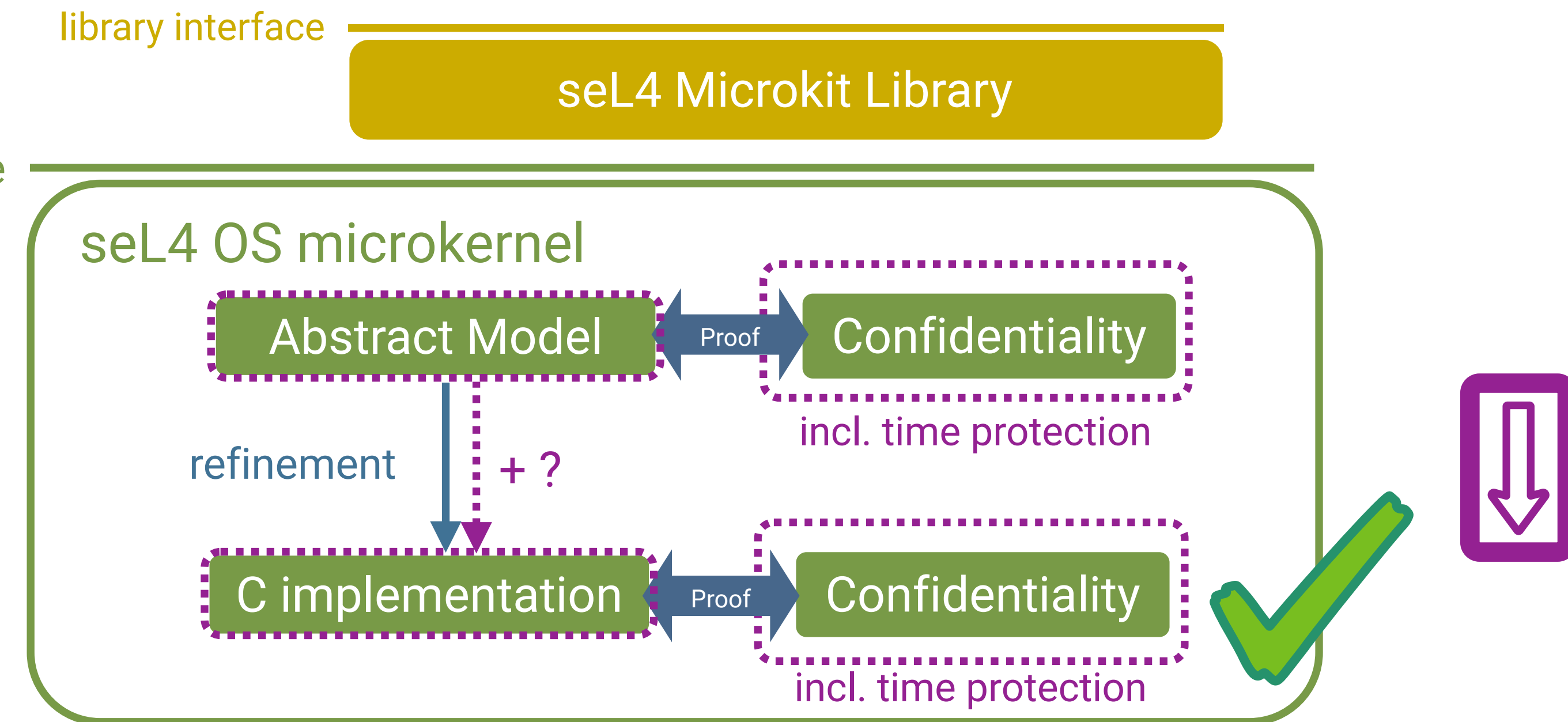






## Microkit-based OS Services

## Time Protection

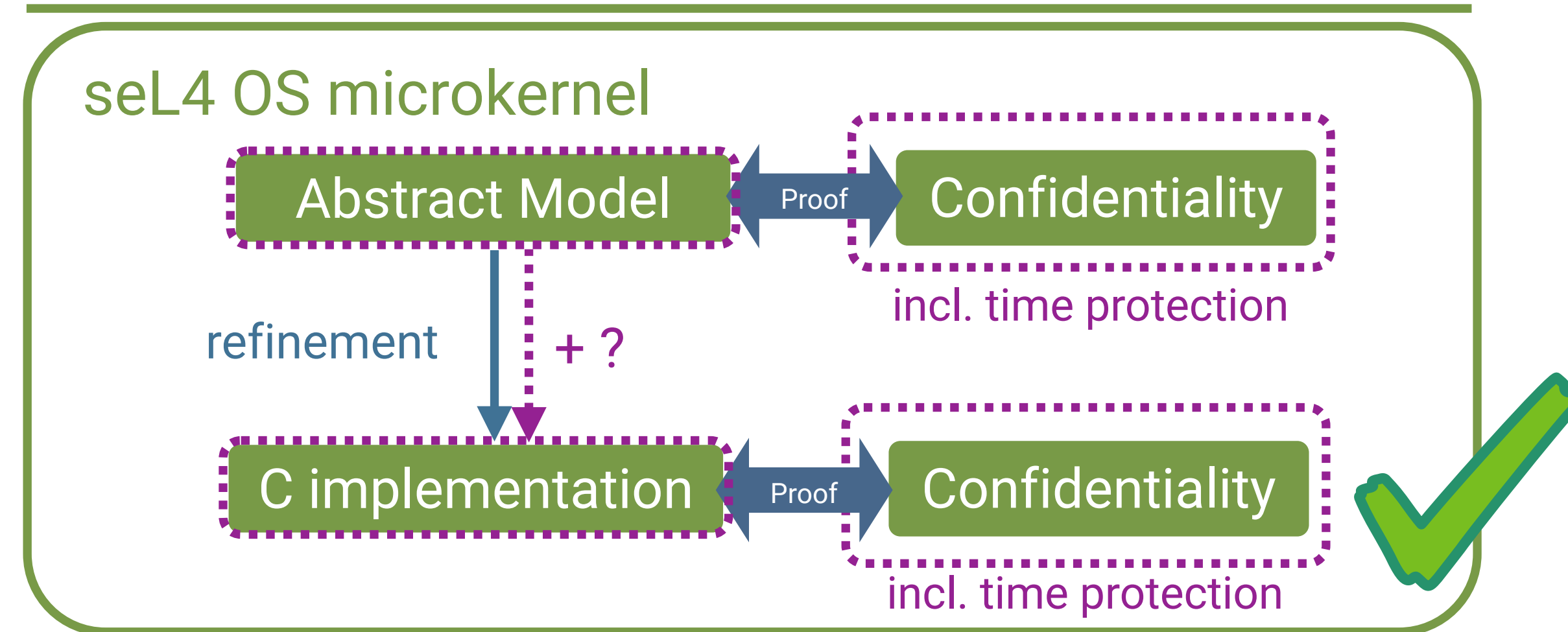


## Microkit-based OS Services

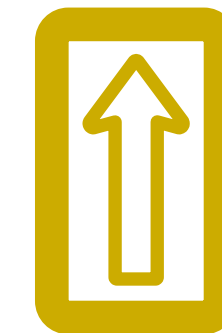
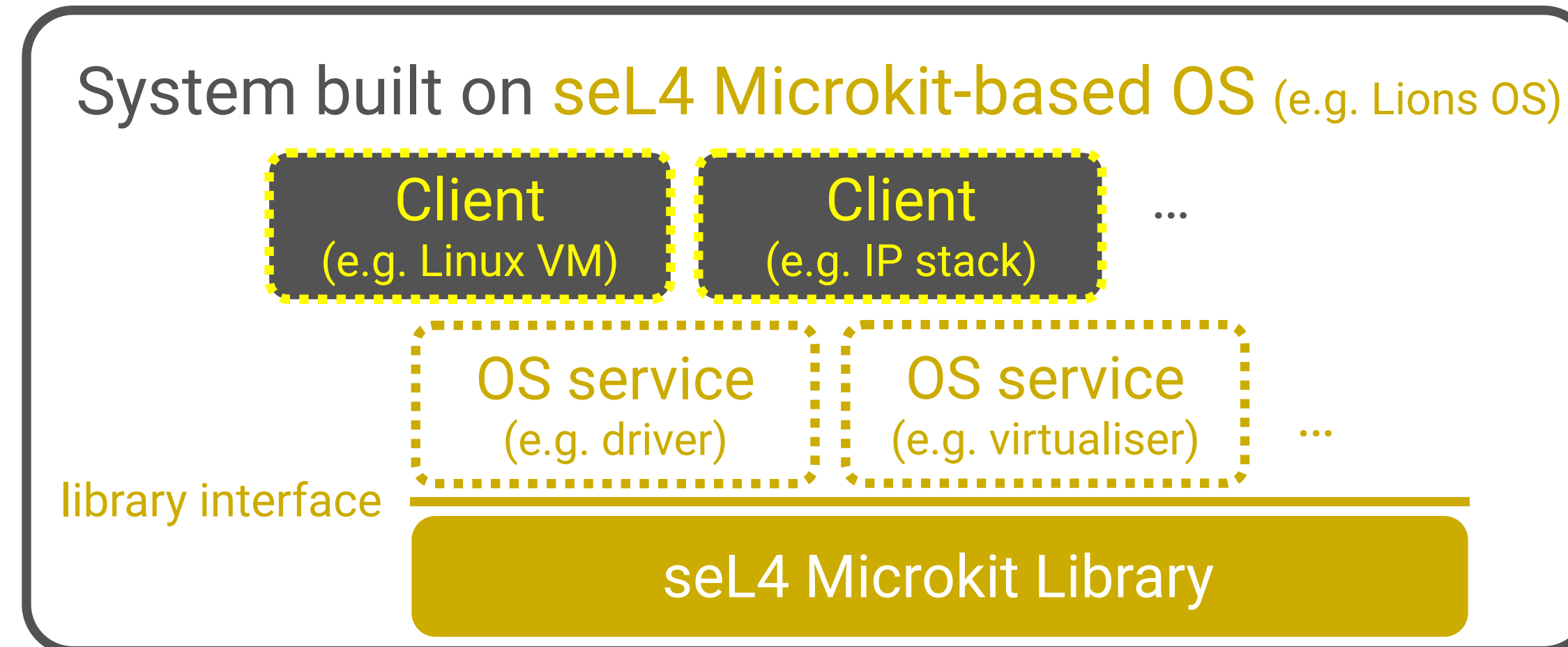


## Time Protection

syscall interface

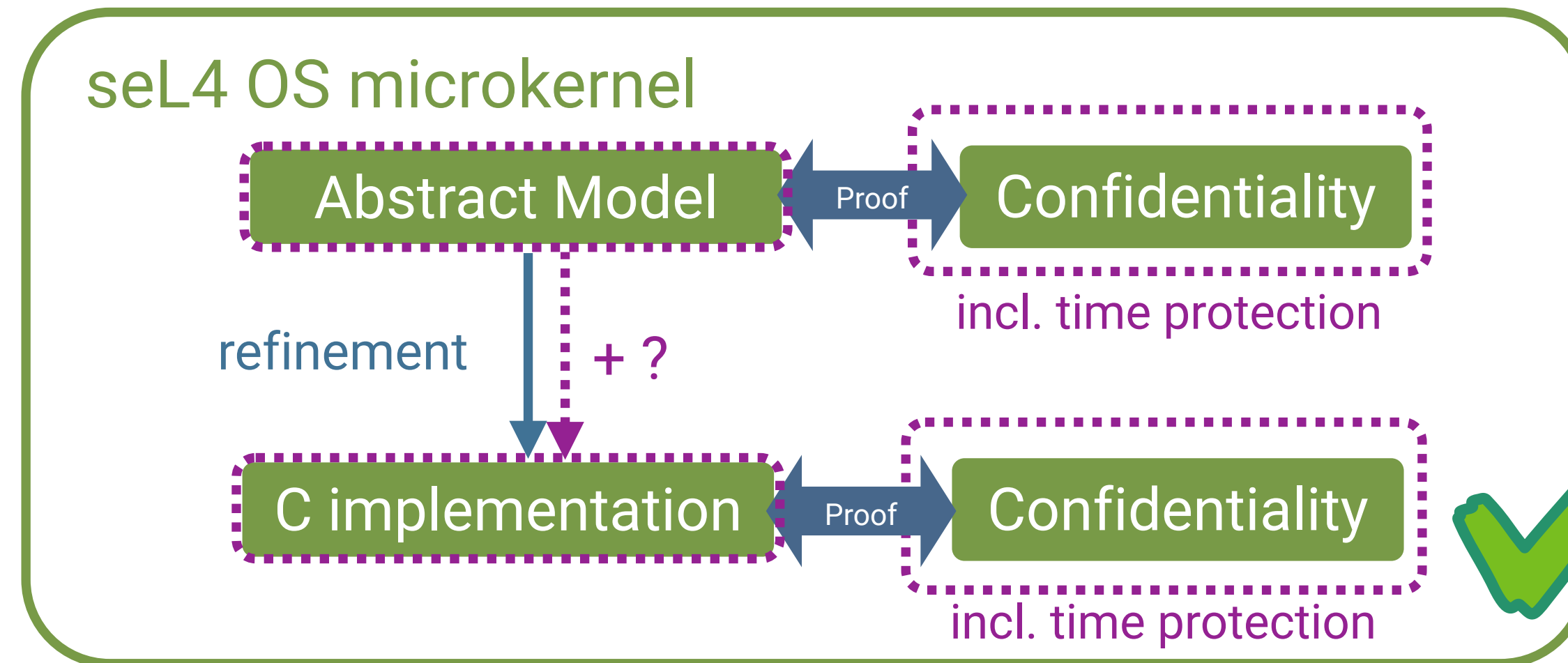


## Microkit-based OS Services

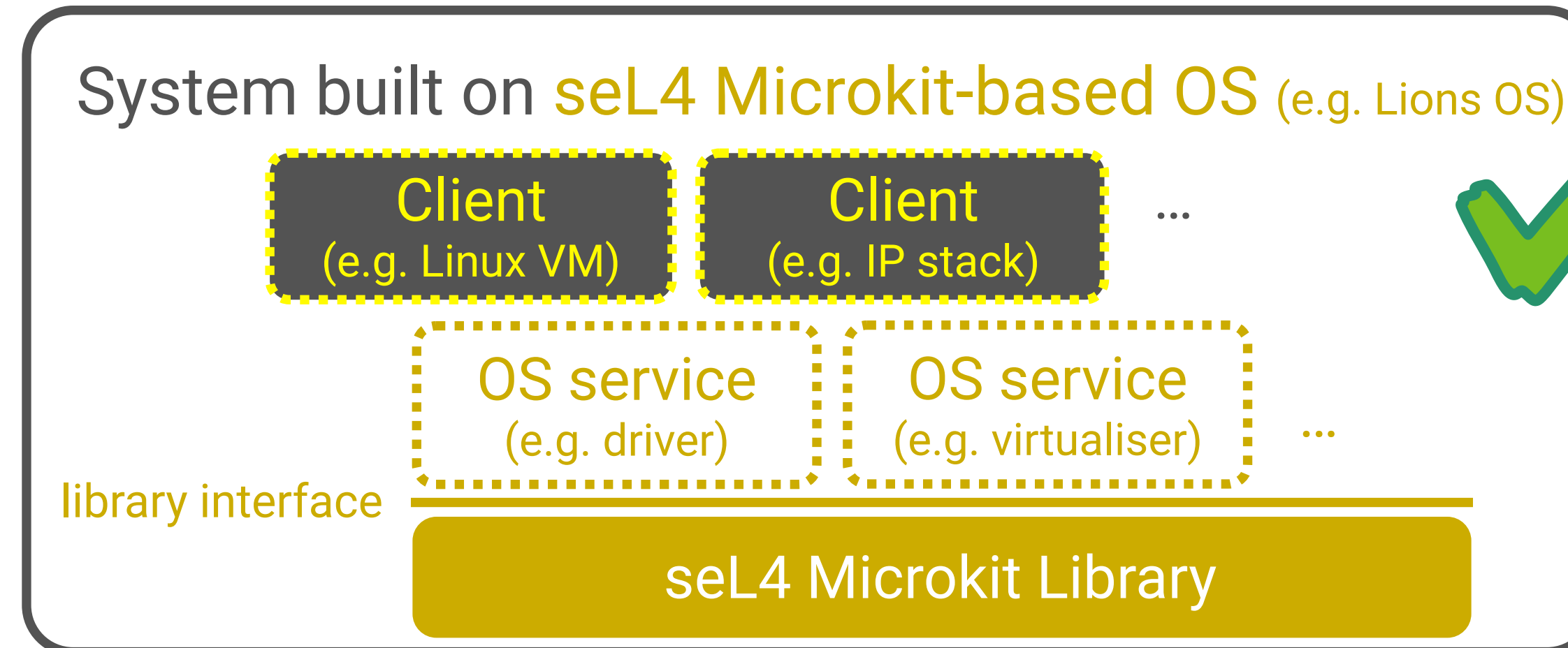


syscall interface

## Time Protection

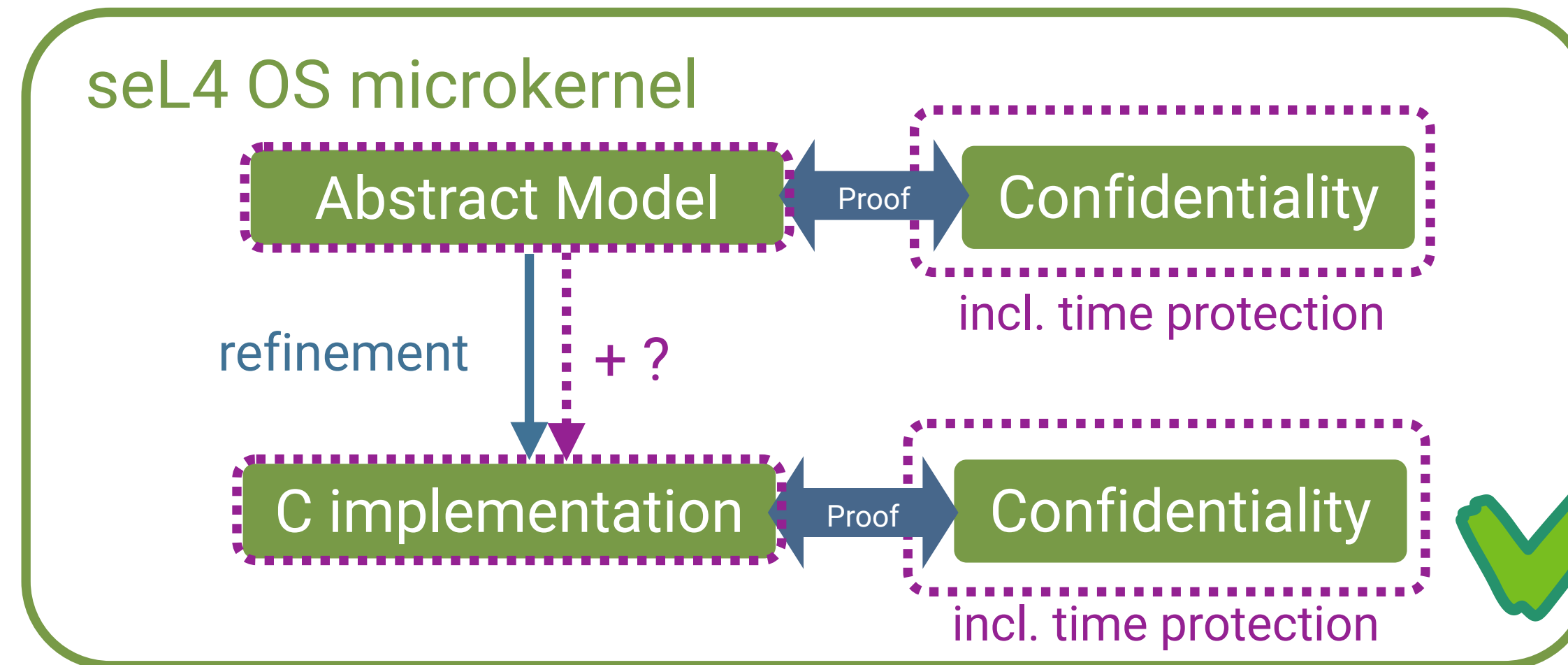


## Microkit-based OS Services

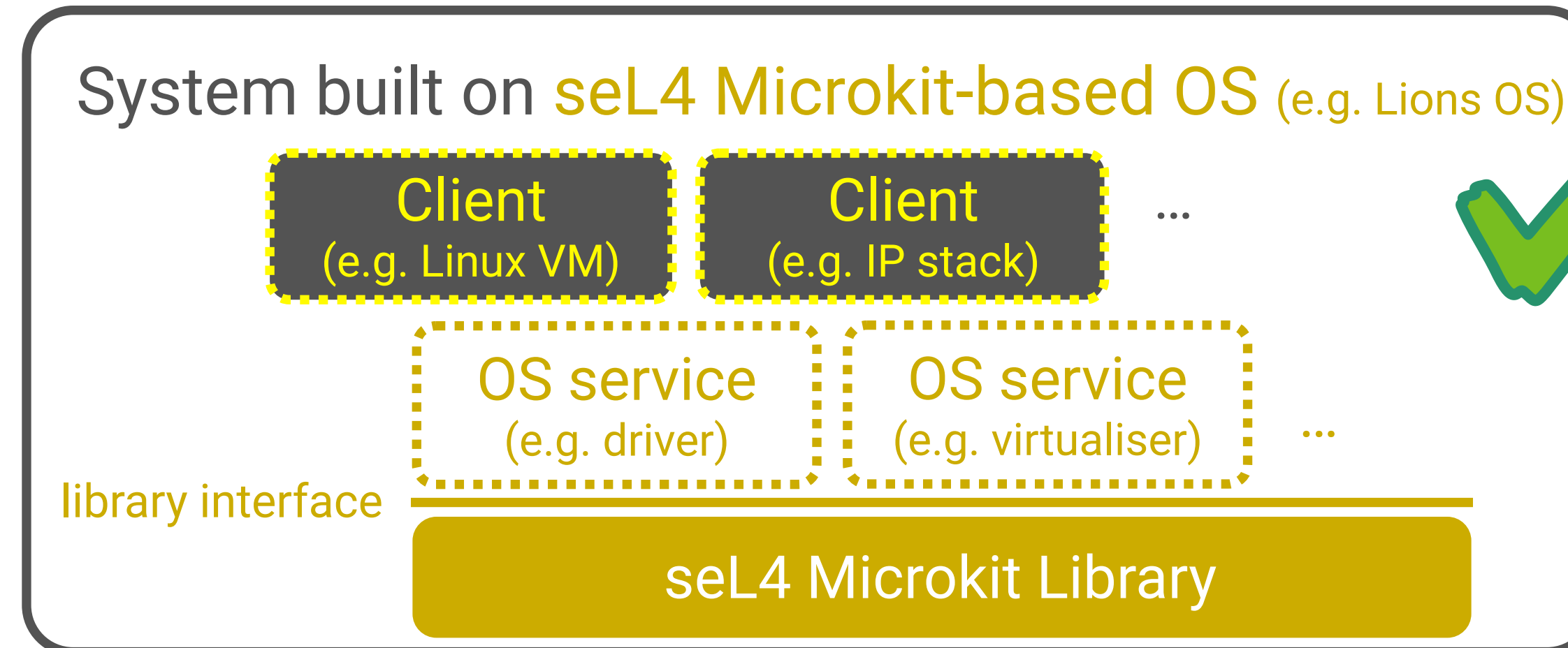


syscall interface

## Time Protection

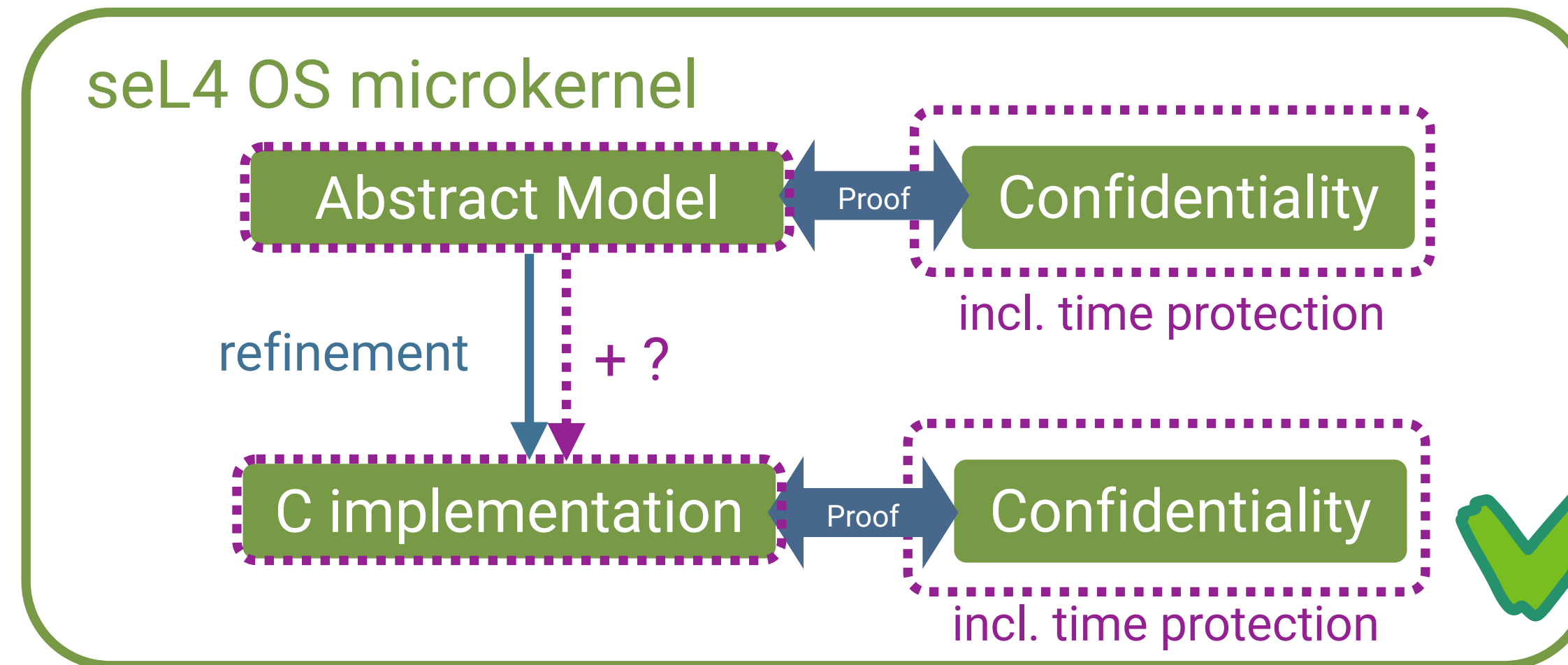


Microkit-based OS Services



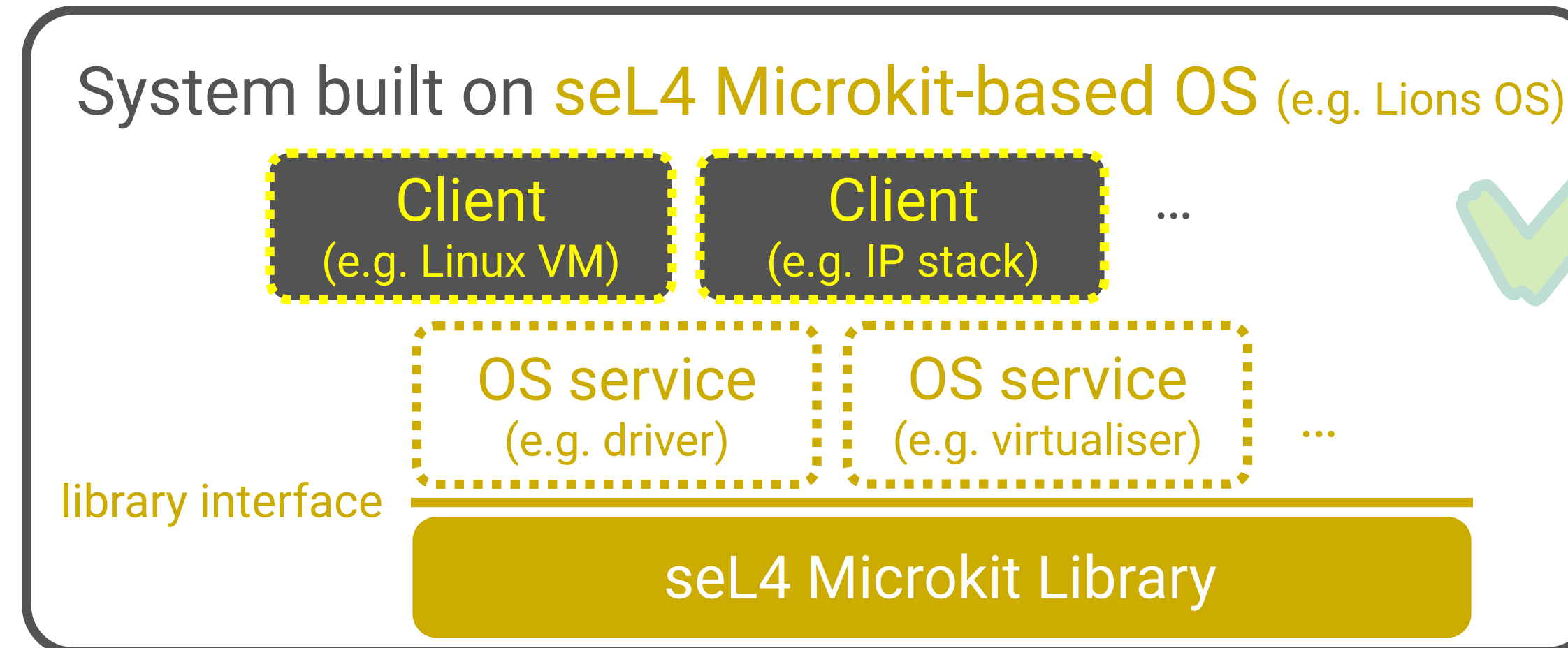
syscall interface

Time Protection



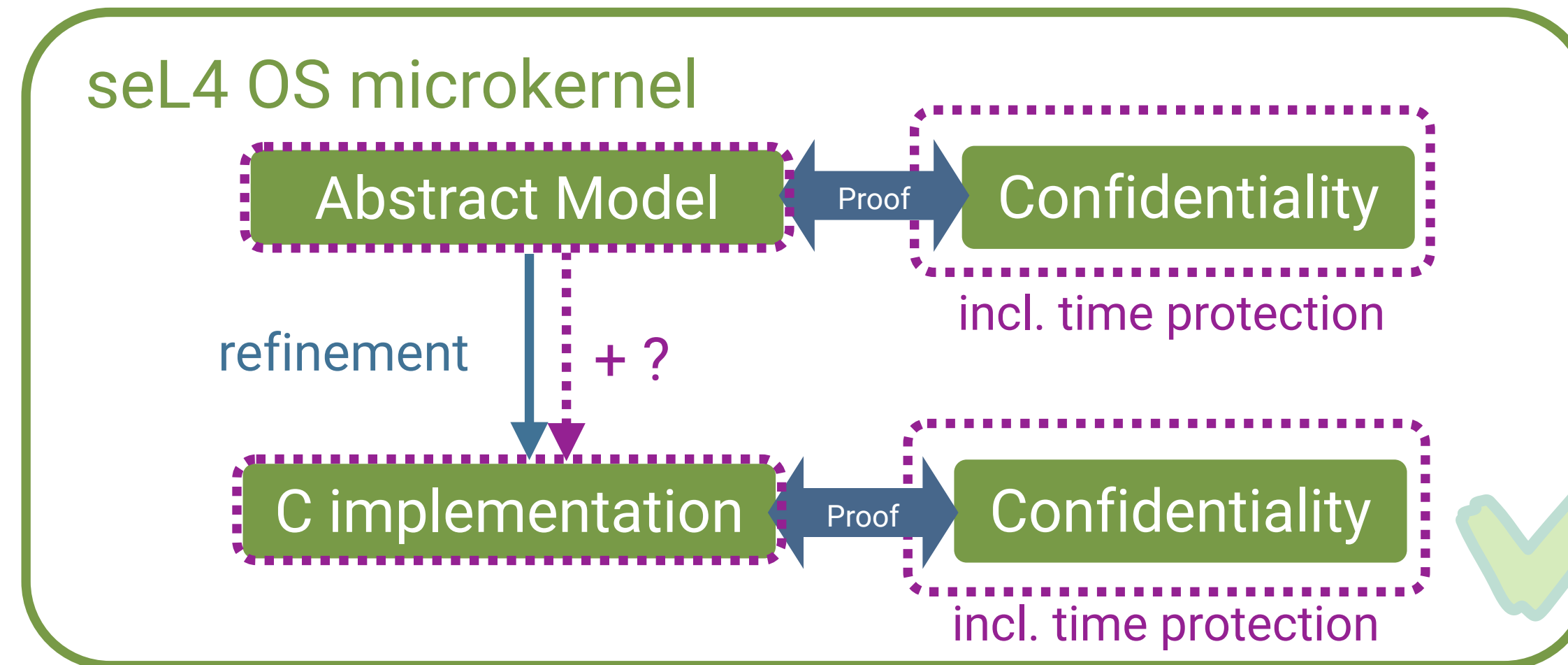
# Verification status

Microkit-based OS Services

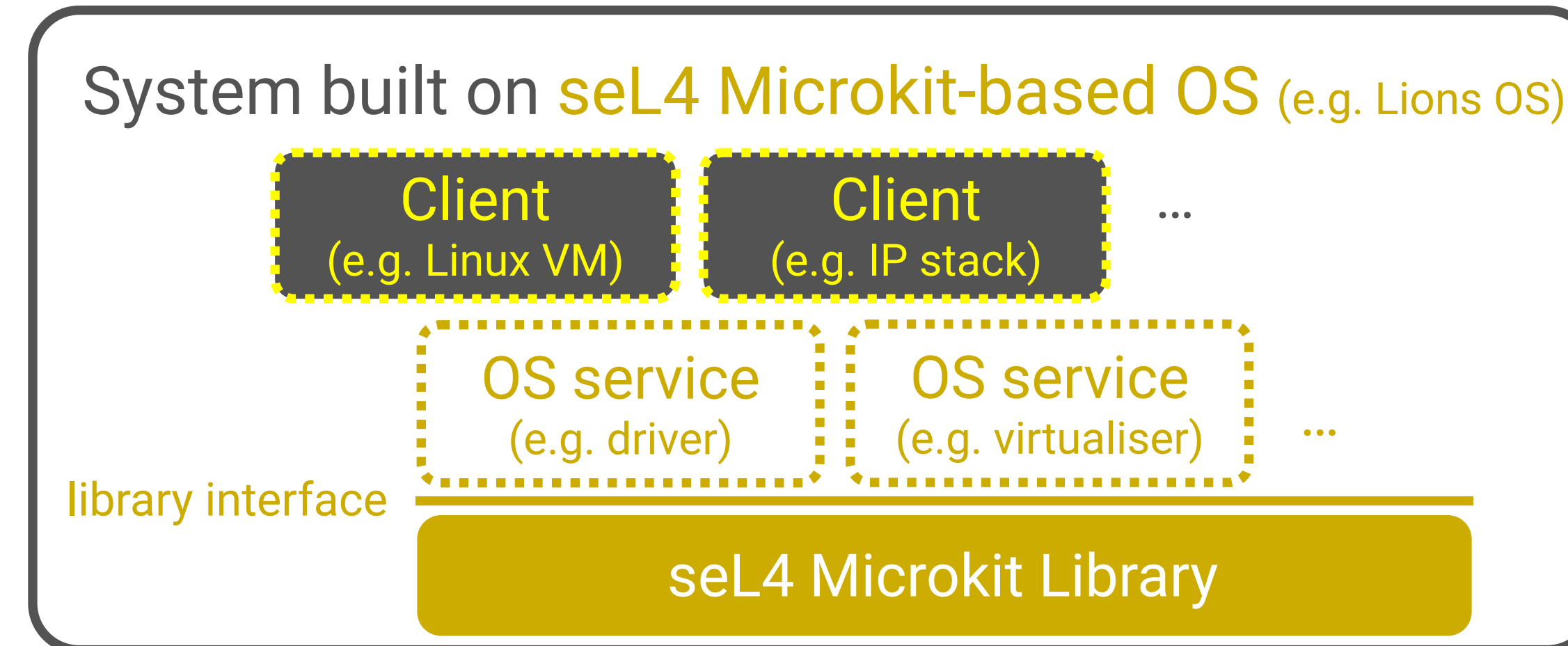


syscall interface

Time Protection

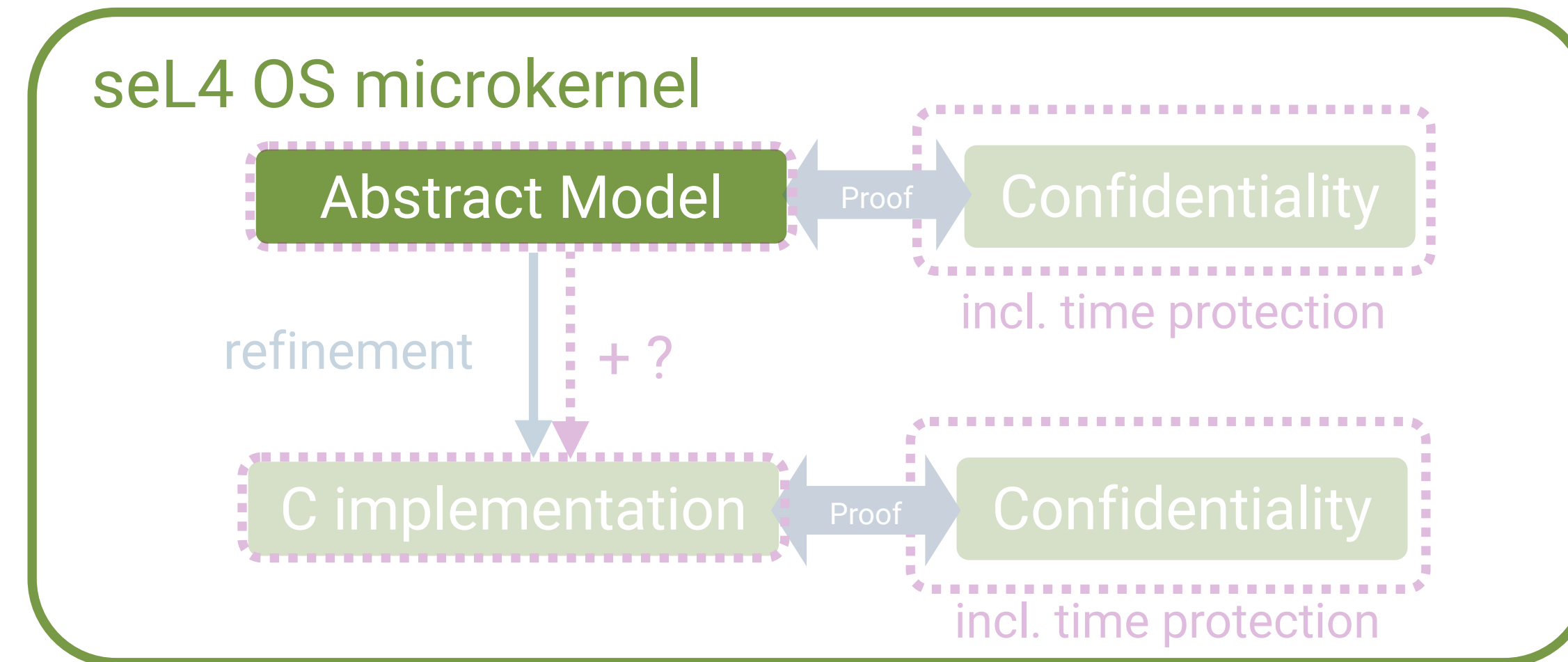


Microkit-based OS Services



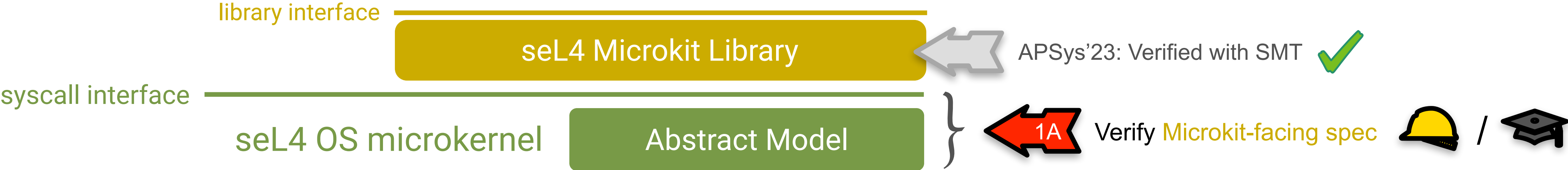
syscall interface

Time Protection





# Microkit-facing OS kernel interface



Challenge: *kernel-perspective vs caller-perspective spec*

# Microkit-facing OS kernel interface



Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑

# Microkit-facing OS kernel interface



## Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑
  - e.g. seL4\_Recv returns immediately

# Microkit-facing OS kernel interface



## Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑
  - e.g. seL4\_Recv returns immediately
  - Prove Hoare triple over single kernel entry/exit

# Microkit-facing OS kernel interface



## Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑
  - e.g. seL4\_Recv returns immediately
  - Prove Hoare triple over single kernel entry/exit
- Blocking cases: Tricky 🎓

# Microkit-facing OS kernel interface



## Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑
  - e.g. seL4\_Recv returns immediately
  - Prove Hoare triple over single kernel entry/exit
- Blocking cases: Tricky 🎓
  - e.g. seL4\_Recv blocks waiting for seL4\_Call

# Microkit-facing OS kernel interface



## Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑
  - e.g. seL4\_Recv returns immediately
  - Prove Hoare triple over single kernel entry/exit
- Blocking cases: Tricky 🎓
  - e.g. seL4\_Recv blocks waiting for seL4\_Call
  - Kernel returns to different user!  
... Caller is woken by seL4\_Call later.

# Microkit-facing OS kernel interface



## Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑
  - e.g. seL4\_Recv returns immediately
  - Prove Hoare triple over single kernel entry/exit
- Blocking cases: Tricky 🎓
  - e.g. seL4\_Recv blocks waiting for seL4\_Call
  - Kernel returns to different user!  
... Caller is woken by seL4\_Call later.
  - **Not handled by prior OS verification work**
    - cf. CertiKOS ESOP'20 - blocks on IO, not another user



# Microkit-facing OS kernel interface

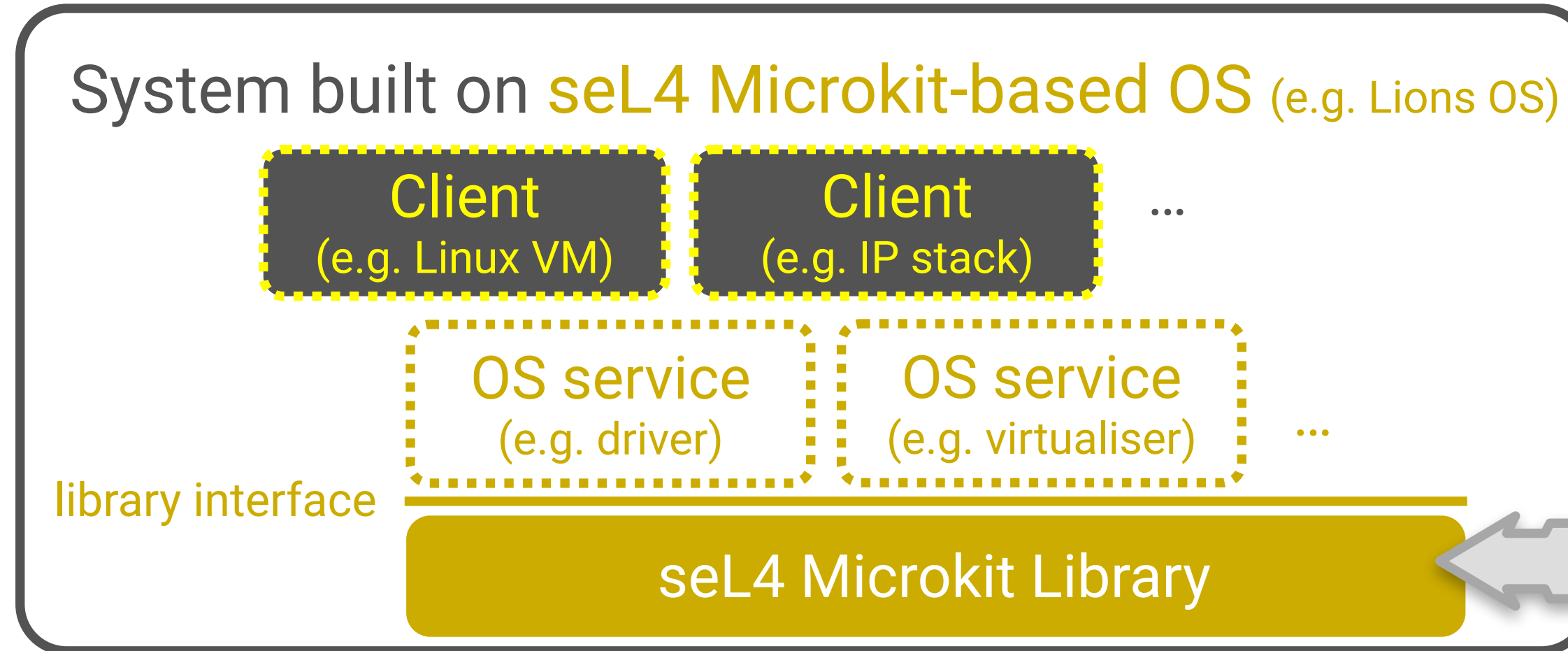


## Challenge: *kernel-perspective vs caller-perspective spec*

- Non-blocking cases: Straightforward 🛑
  - e.g. seL4\_Recv returns immediately
  - Prove Hoare triple over single kernel entry/exit
- Blocking cases: Tricky 🎓
  - e.g. seL4\_Recv blocks waiting for seL4\_Call
  - Kernel returns to different user!  
... Caller is woken by seL4\_Call later.
  - **Not handled by prior OS verification work**
    - cf. CertiKOS ESOP'20 - blocks on IO, not another user



## Microkit-based OS Services

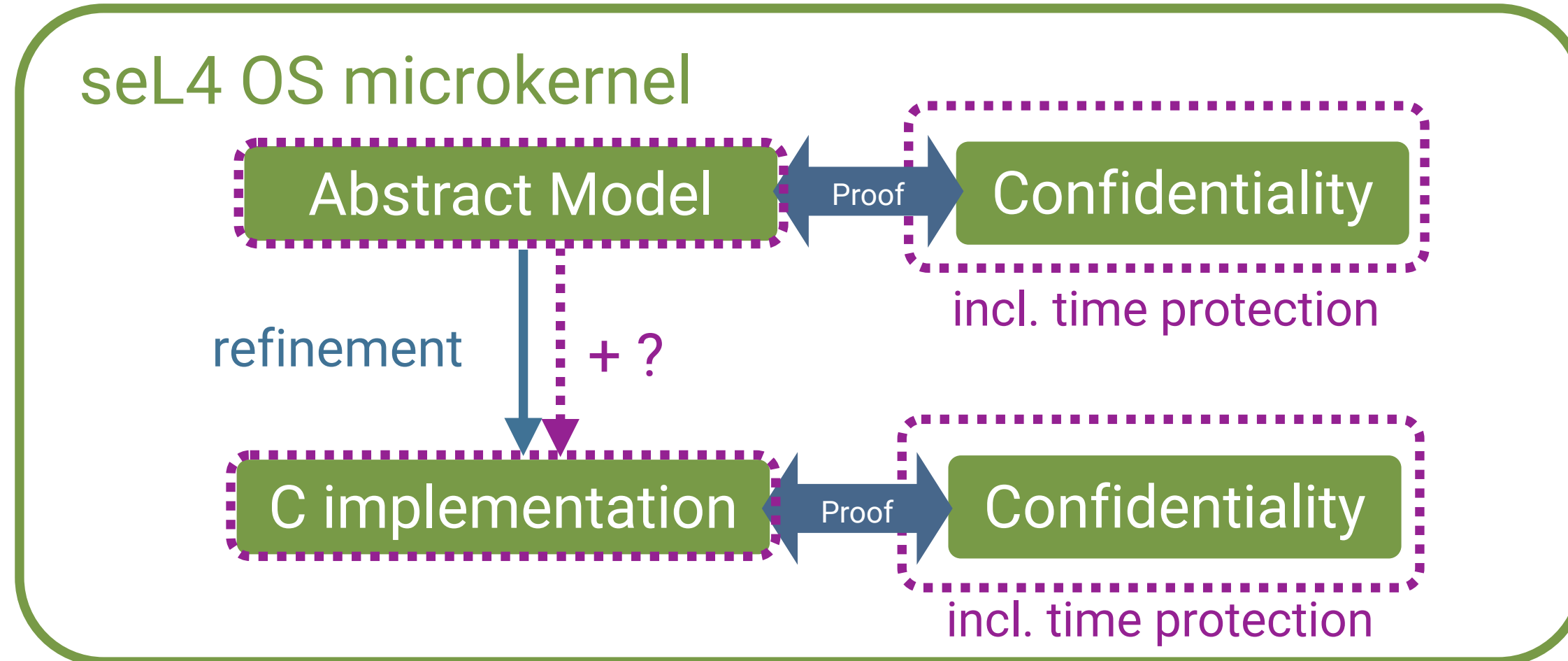


APSys'23: Verified with SMT ✓

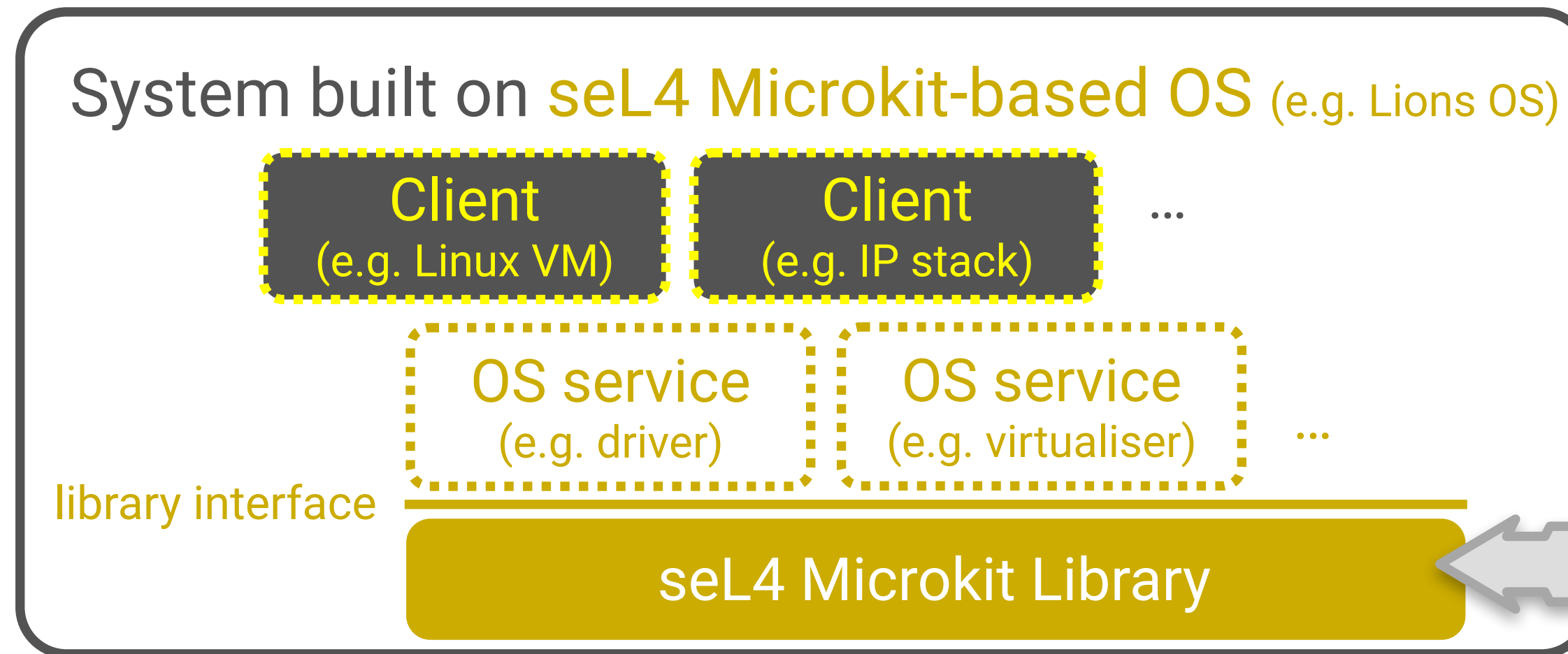
syscall interface

1A Verify Microkit-facing spec 🛑 / 🎓

## Time Protection



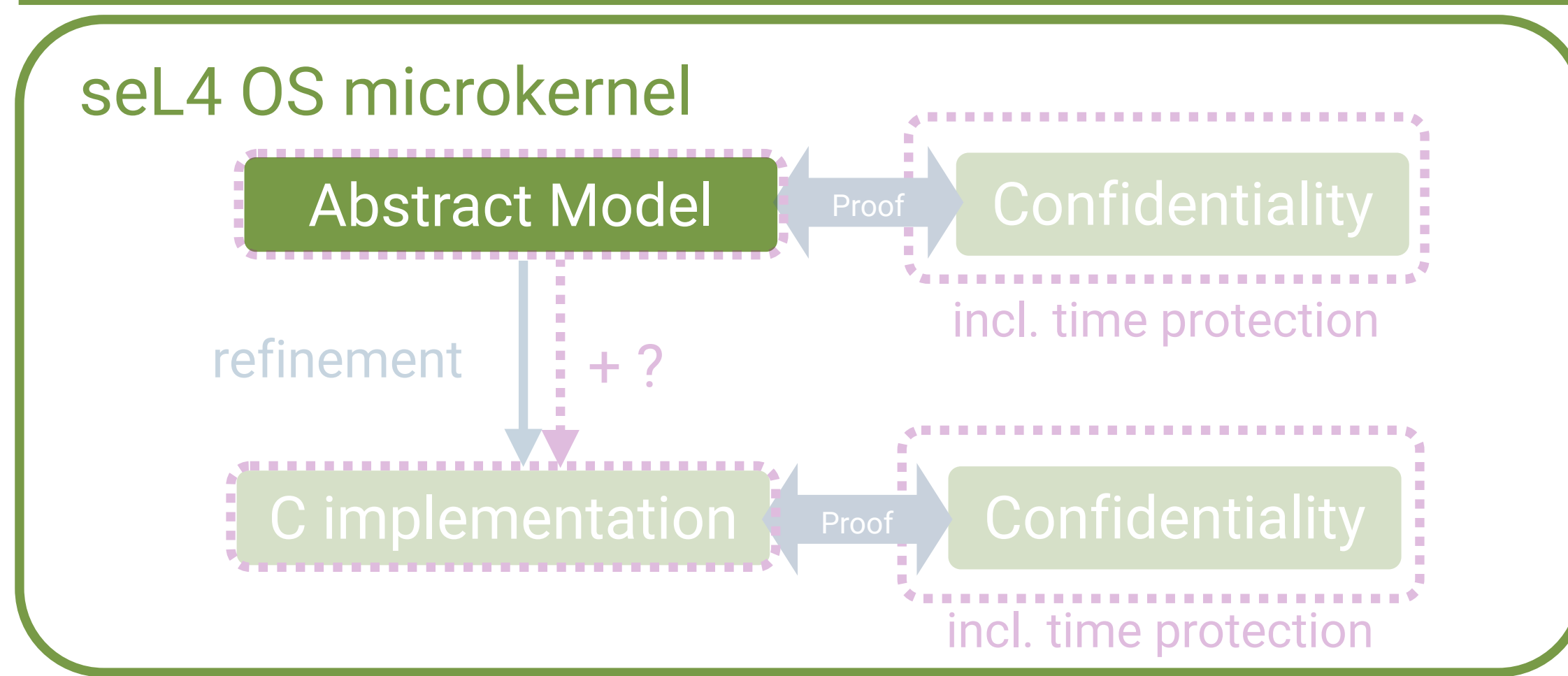
## Microkit-based OS Services



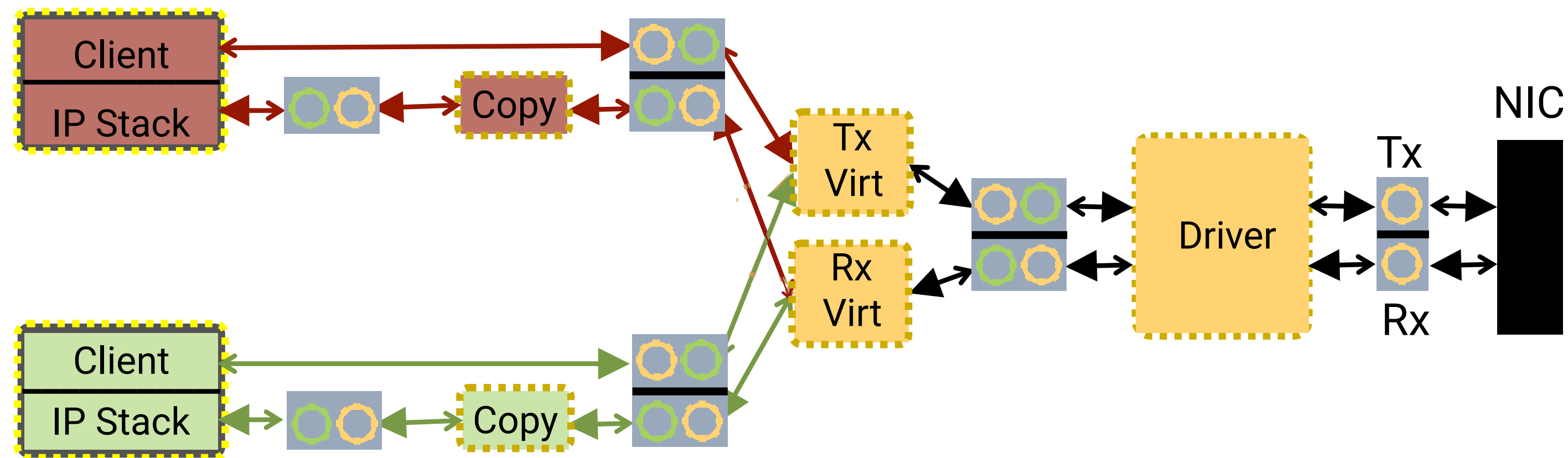
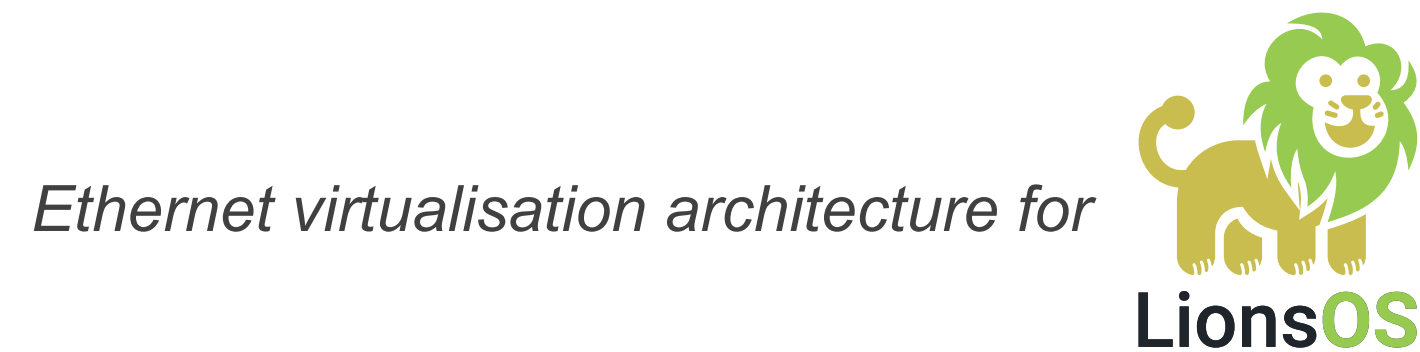
APSys'23: Verified with SMT ✓

1A Verify Microkit-facing spec 🛑 / 🎓

## Time Protection



# Microkit-based OS services, drivers (+ devices)



library interface

seL4 Microkit Library

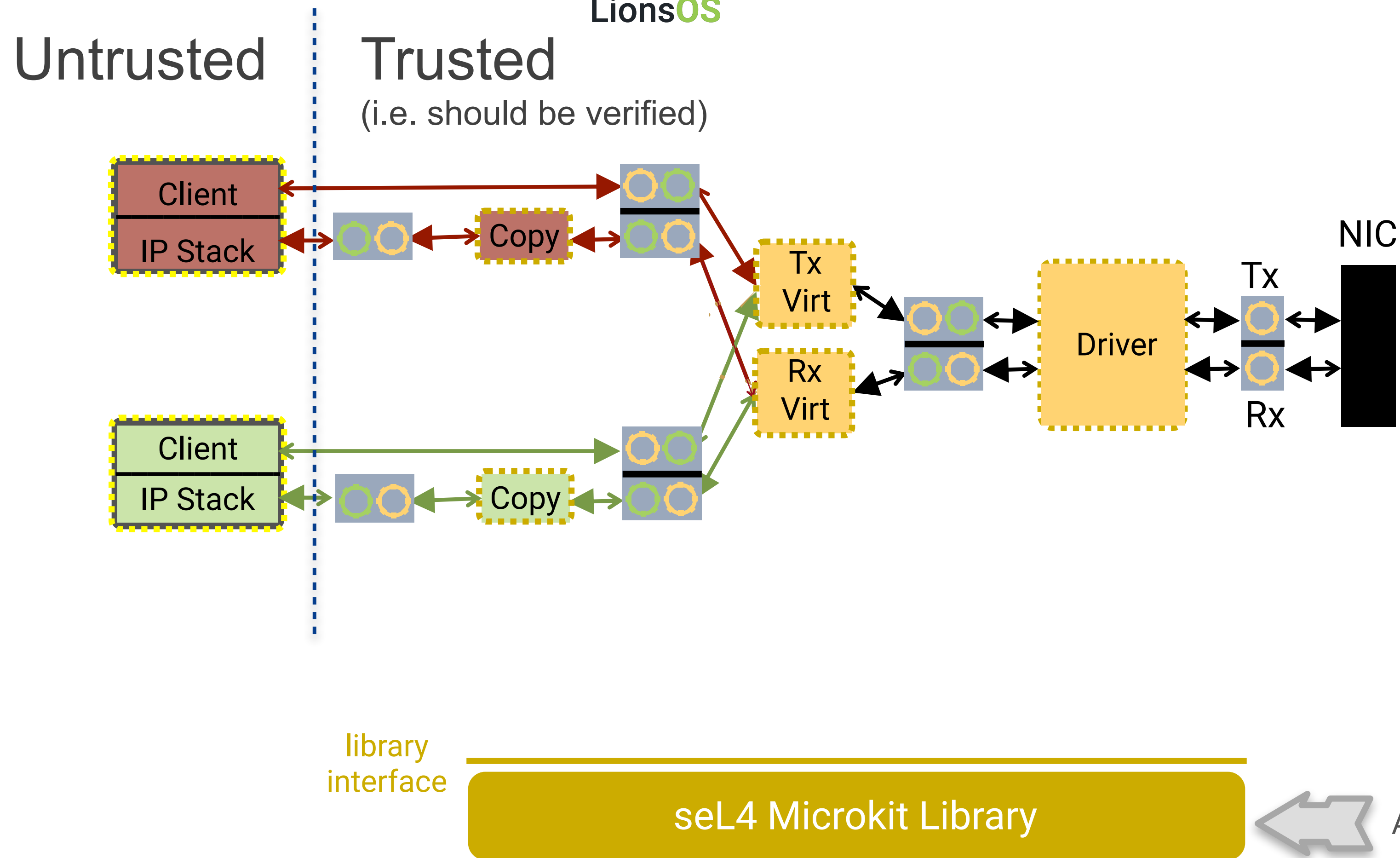
APSys'23: Verified with SMT



# Microkit-based OS services, drivers (+ devices)




Ethernet virtualisation architecture for   
LionsOS

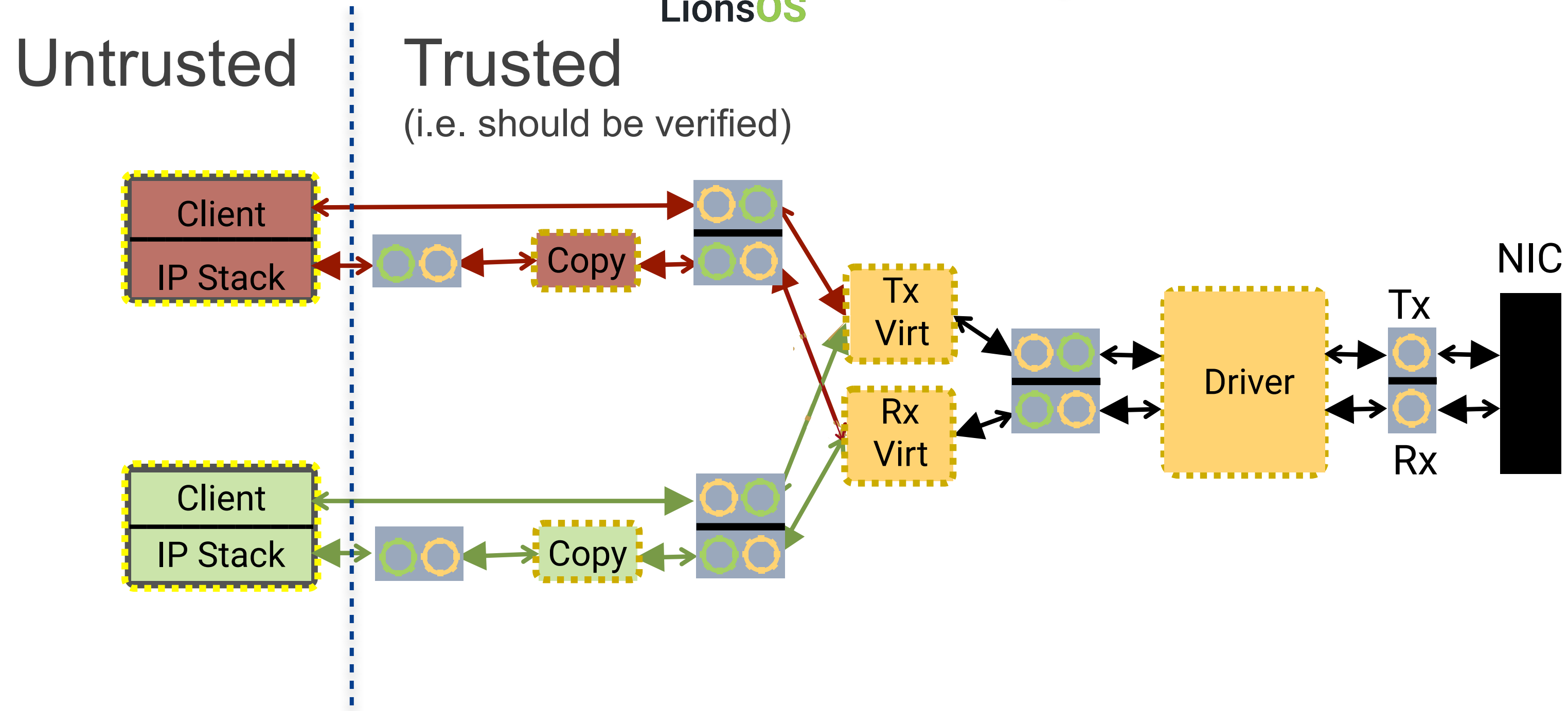


# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Verify inter-service/client communication protocols  / 



library interface

seL4 Microkit Library

APSys'23: Verified with SMT 

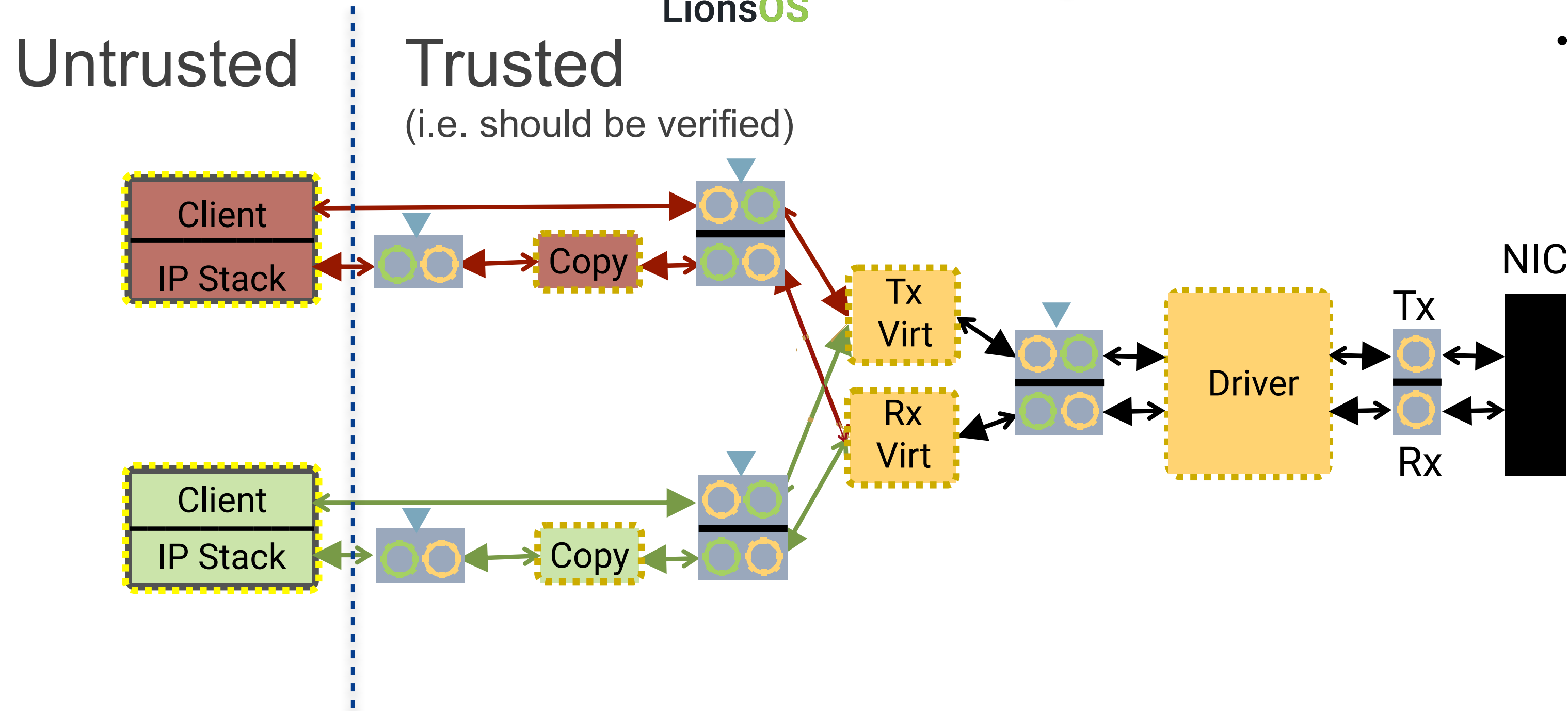
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Verify inter-service/client communication protocols  / 

- Target: SPSC queues



library interface

seL4 Microkit Library

APSys'23: Verified with SMT 

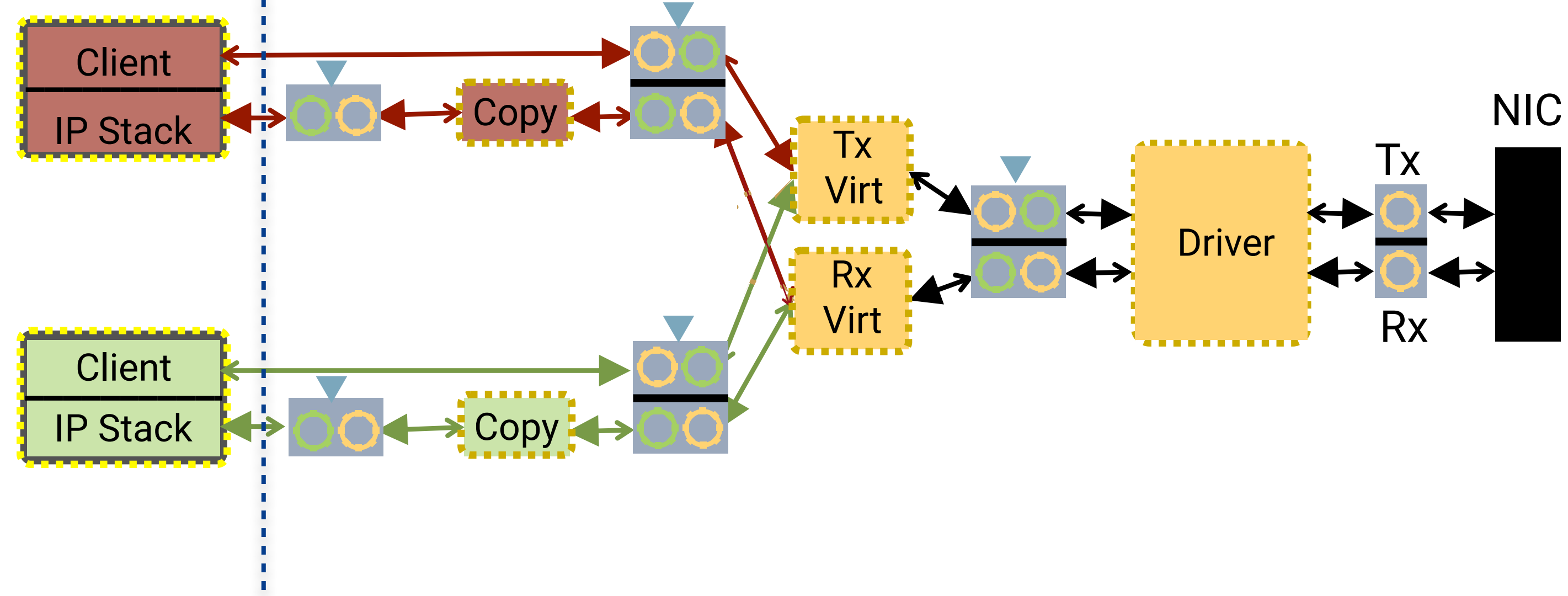
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



Verify inter-service/client communication protocols



Model checking (SPIN)

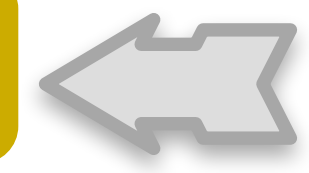
- Target: SPSC queues

- Deadlock freedom for aggressive optimisations



library interface

seL4 Microkit Library



APSys'23: Verified with SMT





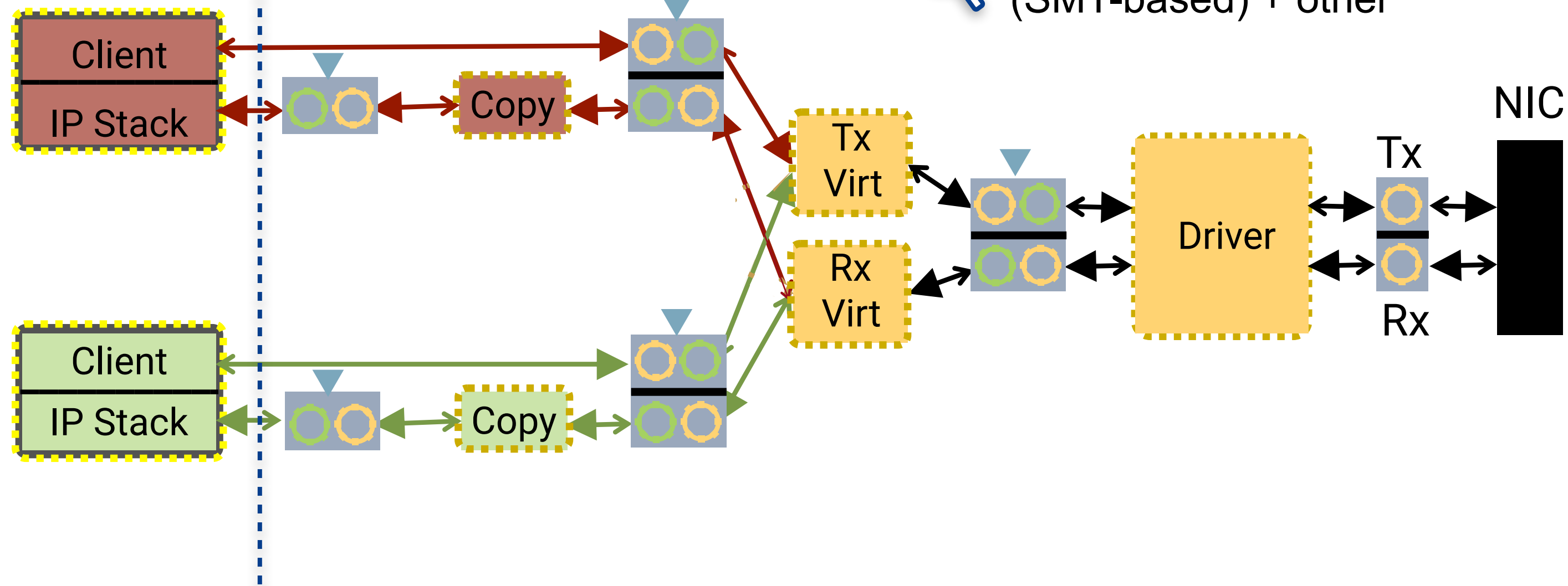
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)





Verify inter-service/client communication protocols  / 

Model checking (SPIN)

Deductive verification (SMT-based) + other

Target: SPSC queues

- Deadlock freedom for aggressive optimisations 
- Correctness under weak memory models 

library interface

seL4 Microkit Library

APSys'23: Verified with SMT 

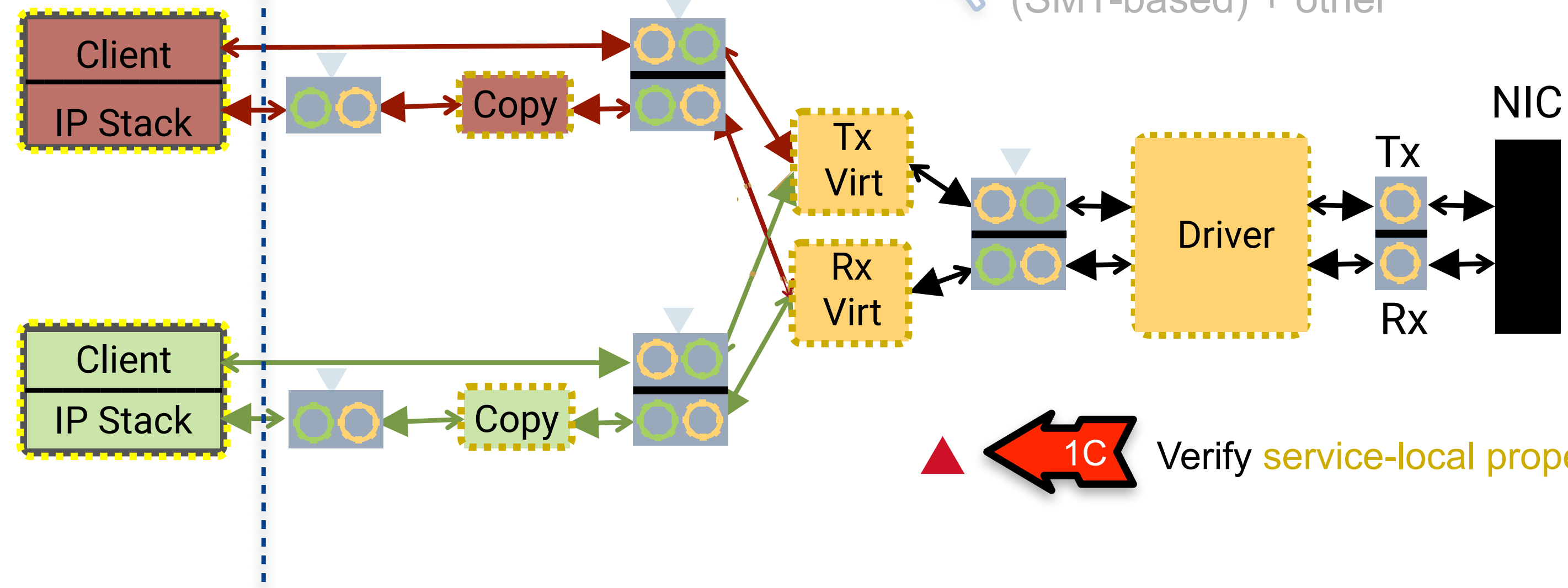
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



1B Verify inter-service/client communication protocols

- Model checking (SPIN)
- Deductive verification (SMT-based) + other



- Target: SPSC queues
- Deadlock freedom for aggressive optimisations
- Correctness under weak memory models

1C Verify service-local properties




library interface

seL4 Microkit Library

APSys'23: Verified with SMT

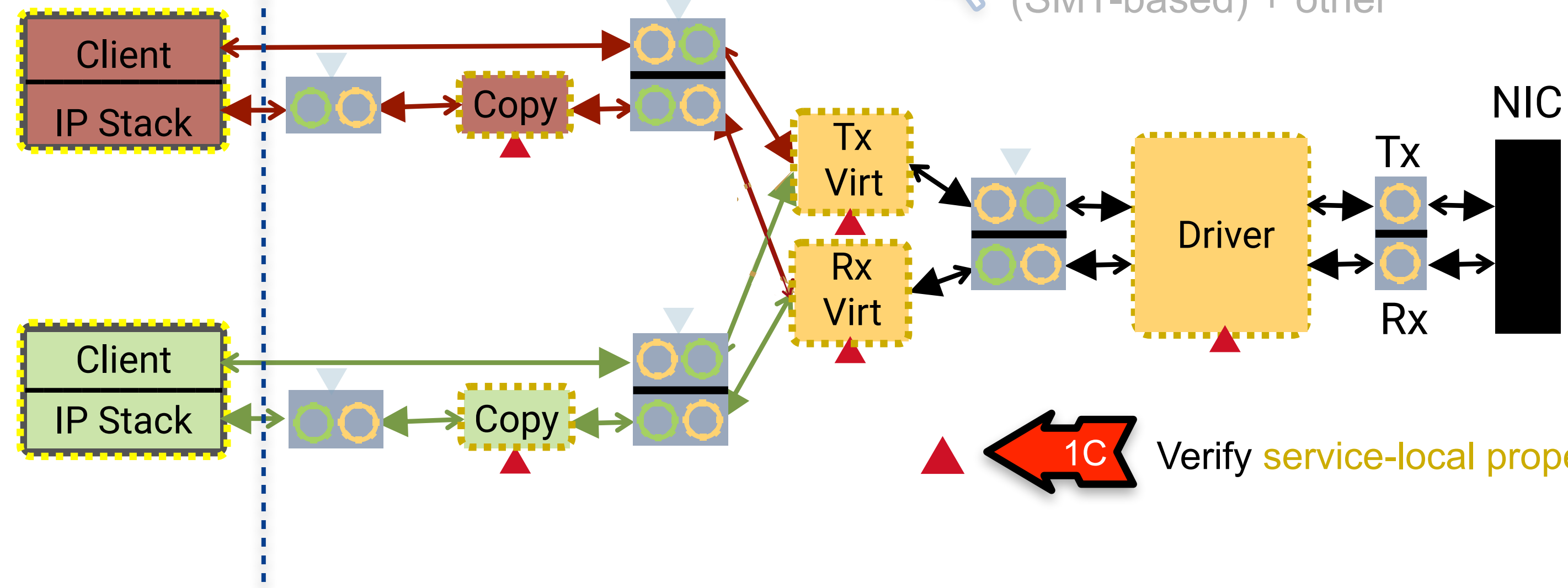
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



1B Verify inter-service/client communication protocols

- Model checking (SPIN)
- Deductive verification (SMT-based) + other



- Target: SPSC queues
- Deadlock freedom for aggressive optimisations
- Correctness under weak memory models

1C Verify service-local properties



- Targets: drivers, components


library interface

seL4 Microkit Library

APSys'23: Verified with SMT

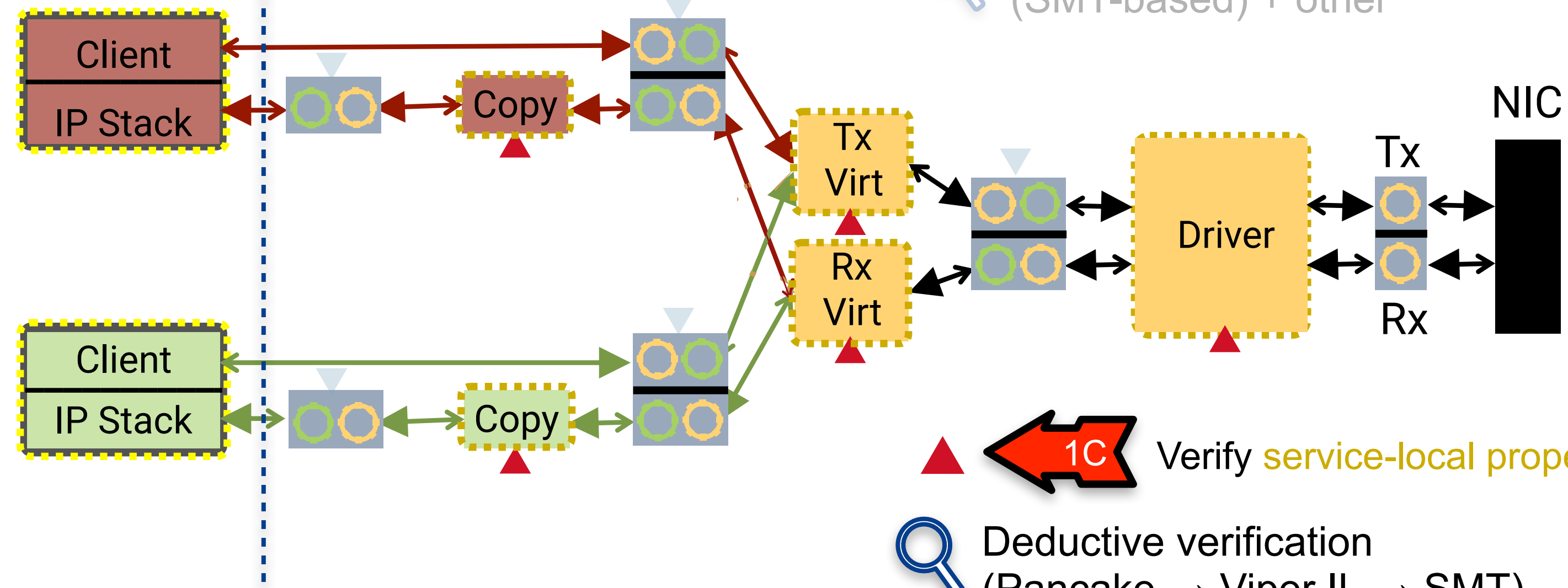
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



1B Verify inter-service/client communication protocols

- Model checking (SPIN)
- Deductive verification (SMT-based) + other



- Target: SPSC queues
- Deadlock freedom for aggressive optimisations
- Correctness under weak memory models

1C Verify service-local properties

- Deductive verification (Pancake → Viper IL → SMT)



- Targets: drivers, components

library interface

seL4 Microkit Library

APSys'23: Verified with SMT

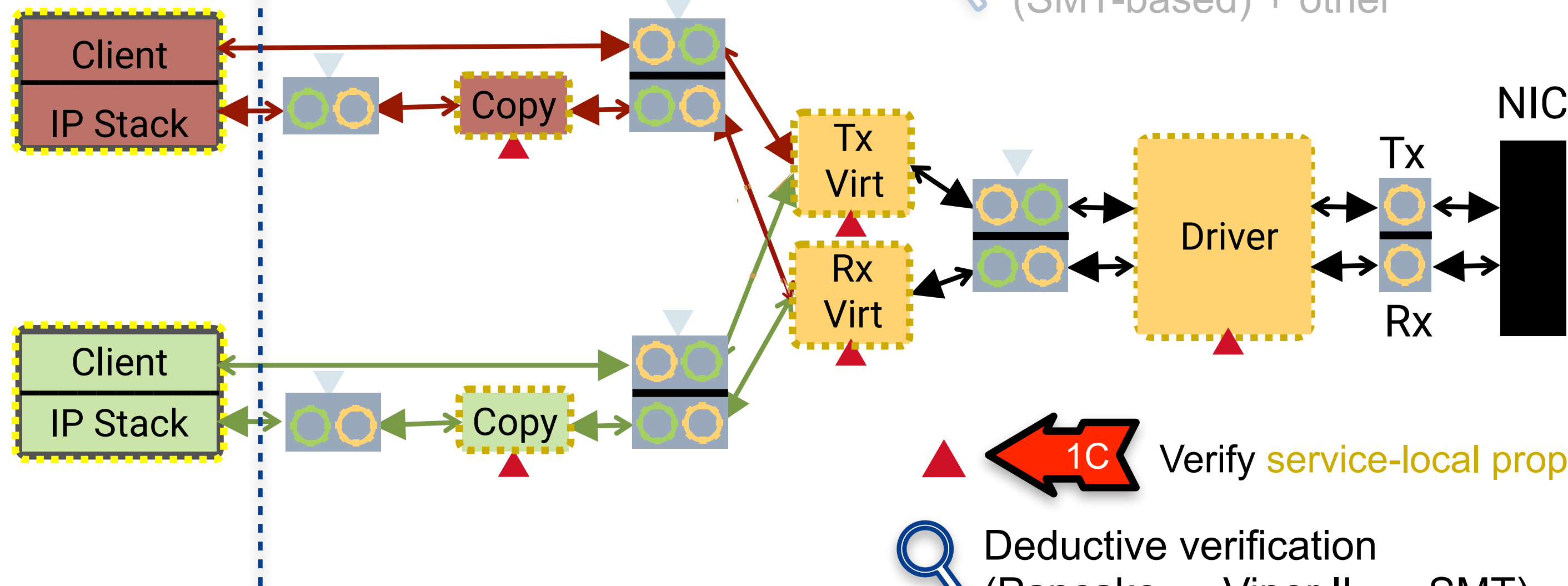
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



1B Verify inter-service/client communication protocols



Model checking (SPIN)

Deductive verification (SMT-based) + other

• Target: SPSC queues

• Deadlock freedom for aggressive optimisations

• Correctness under weak memory models

1C Verify service-local properties



Deductive verification (Pancake → Viper IL → SMT)

Interactive theorem proving (HOL4)

• Targets: drivers, components

library interface

seL4 Microkit Library



APSys'23: Verified with SMT



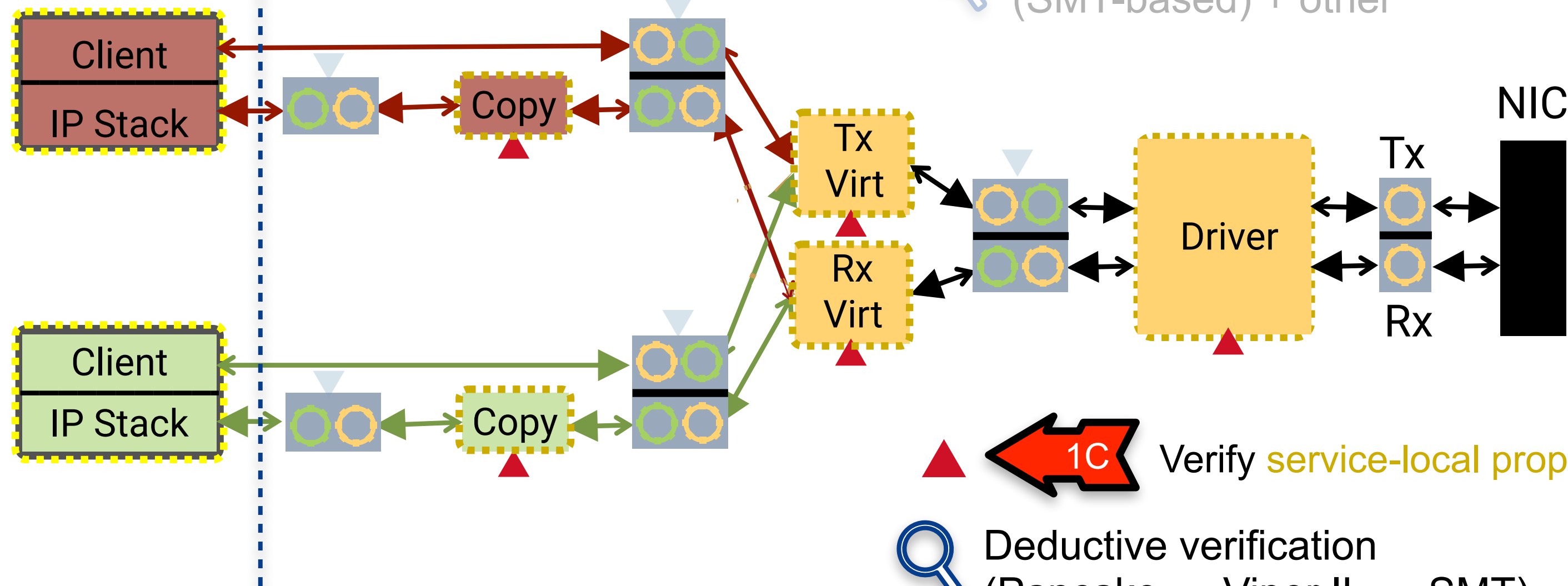
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



1B Verify inter-service/client communication protocols



Model checking (SPIN)

Deductive verification (SMT-based) + other

Target: SPSC queues

- Deadlock freedom for aggressive optimisations
- Correctness under weak memory models



1C Verify service-local properties



Deductive verification (Pancake → Viper IL → SMT)

Interactive theorem proving (HOL4)

Targets: drivers, components

- Functional correctness

library interface

seL4 Microkit Library

APSys'23: Verified with SMT

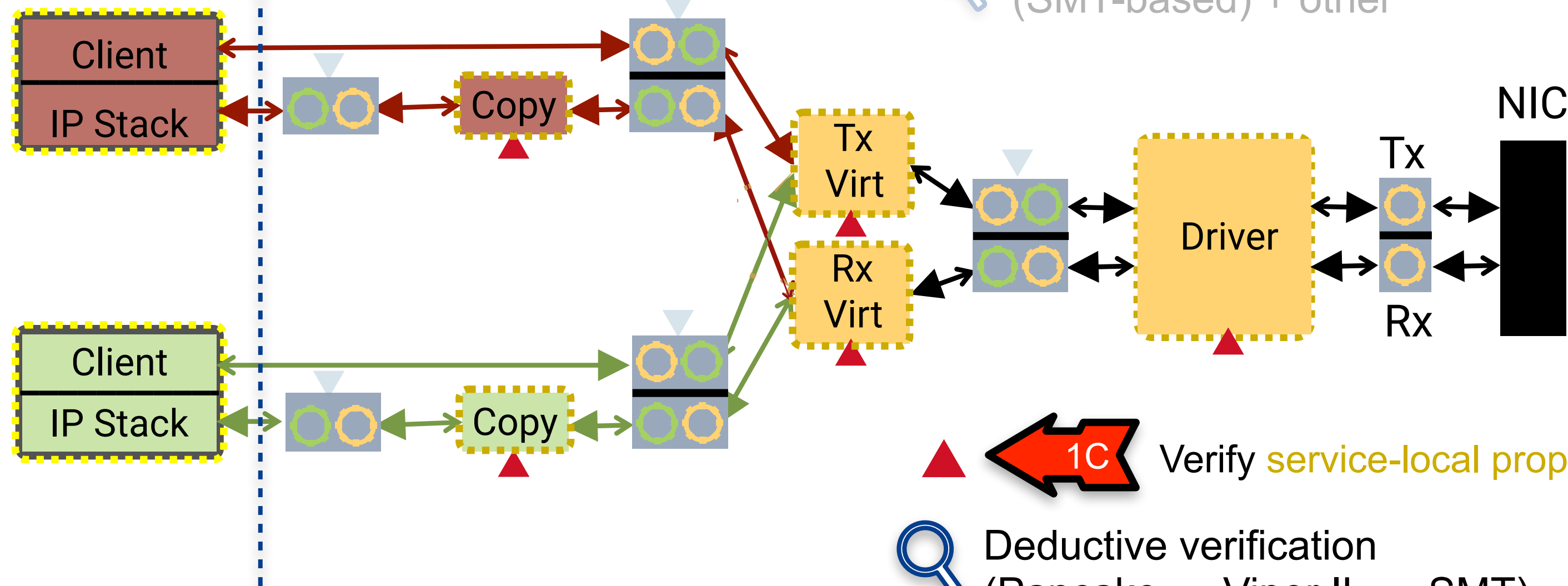
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



1B Verify inter-service/client communication protocols



Model checking (SPIN)

Deductive verification (SMT-based) + other

Target: SPSC queues

- Deadlock freedom for aggressive optimisations
- Correctness under weak memory models



1C Verify service-local properties



Deductive verification (Pancake → Viper IL → SMT)

Interactive theorem proving (HOL4)

Targets: drivers, components

- Functional correctness
- Requirements for device verification



library interface

seL4 Microkit Library

APSys'23: Verified with SMT

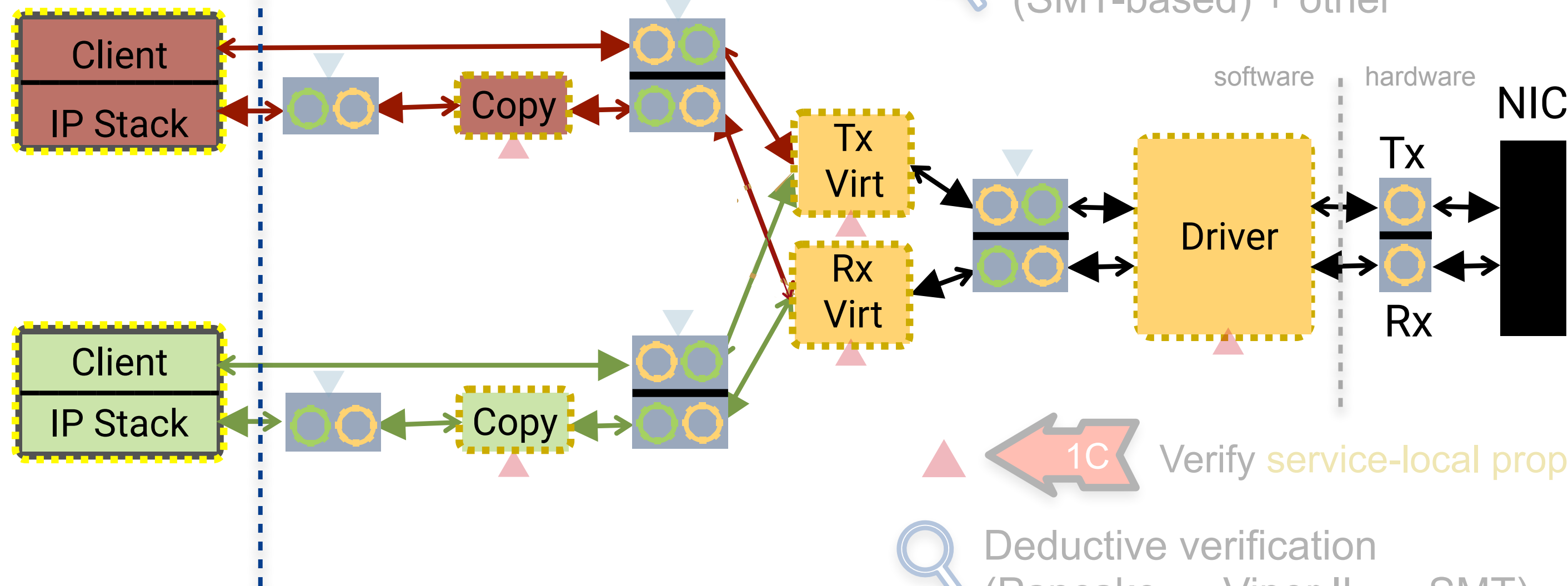
# Microkit-based OS services, drivers (+ devices)



Ethernet virtualisation architecture for  LionsOS

Untrusted

Trusted  
(i.e. should be verified)



1B Verify inter-service/client communication protocols



Model checking (SPIN)

Deductive verification (SMT-based) + other

• Target: SPSC queues

• Deadlock freedom for aggressive optimisations

• Correctness under weak memory models



1C Verify service-local properties



Deductive verification (Pancake → Viper IL → SMT)

Interactive theorem proving (HOL4)

• Targets: drivers, components

• Functional correctness

• Requirements for device verification



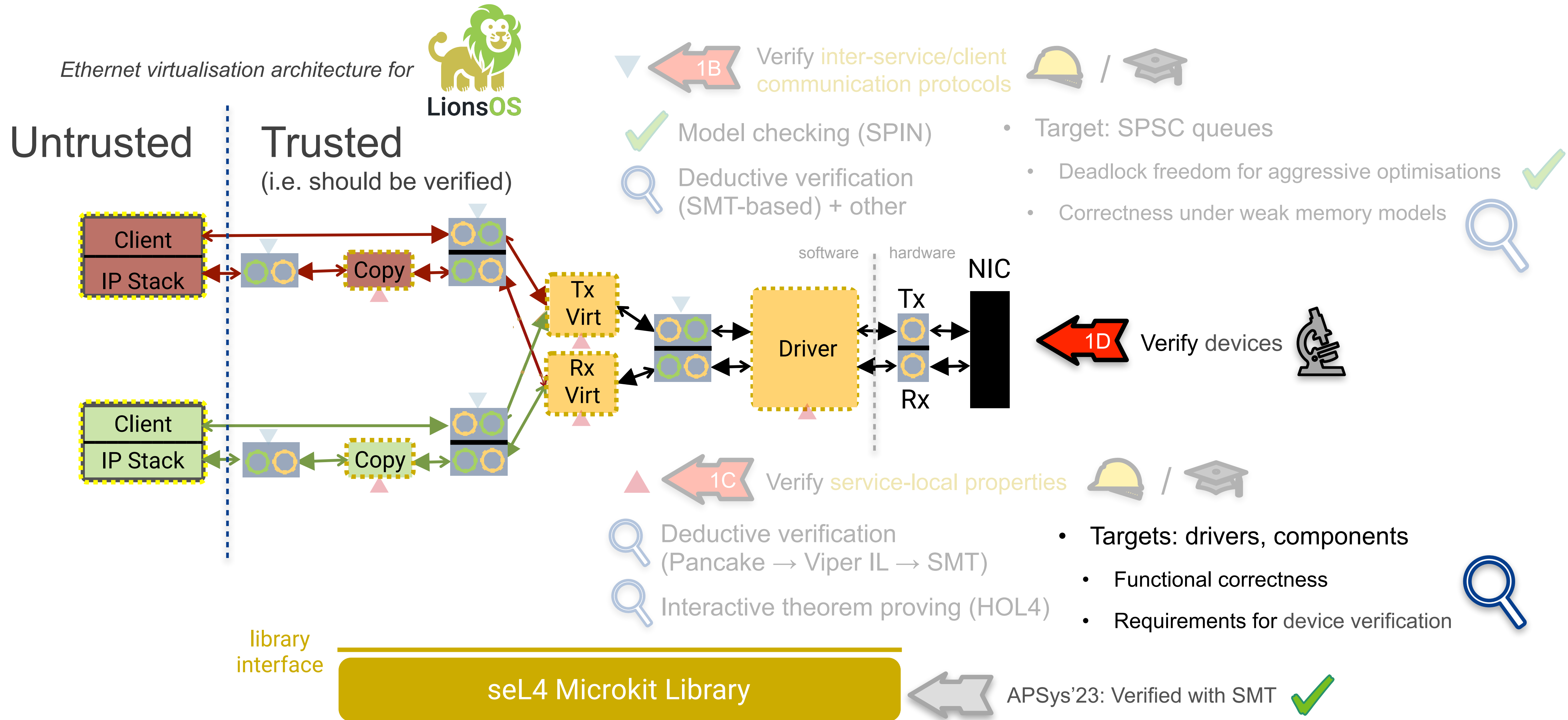
library interface

seL4 Microkit Library

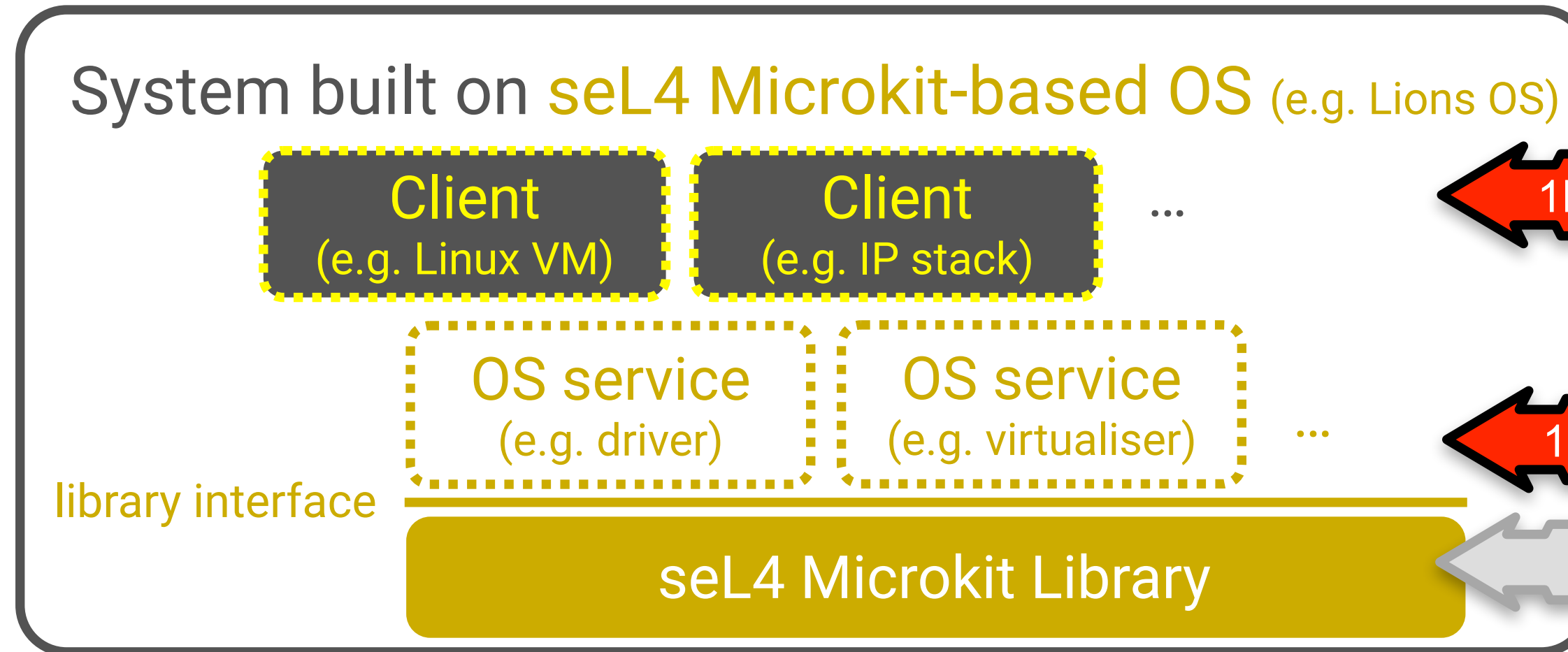
APSys'23: Verified with SMT



# Microkit-based OS services, drivers (+ devices)

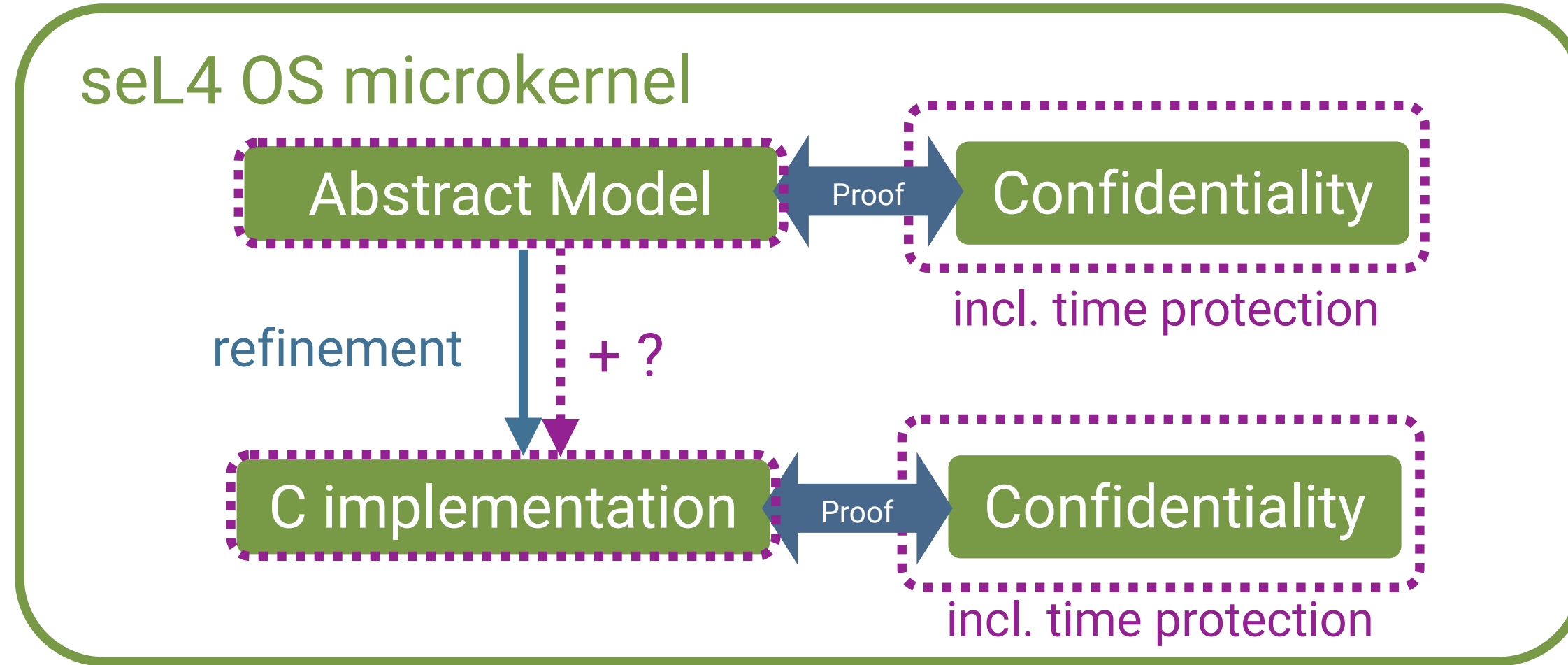


## Microkit-based OS Services

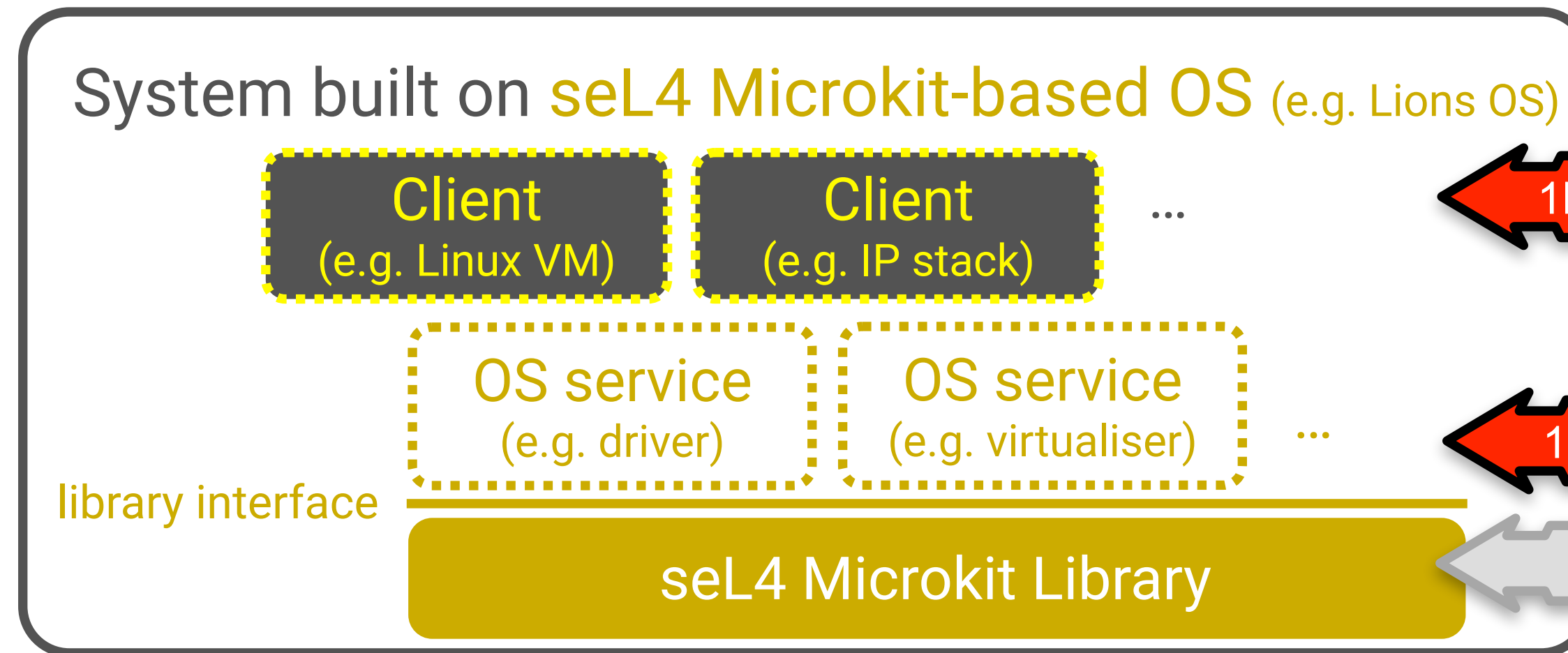


- ← 1B Verify inter-service/client communication protocols /
- ← 1D Verify devices
- ← 1C Verify service-local properties /
- ← APSys'23: Verified with SMT
- ← 1A Verify Microkit-facing spec /

## Time Protection

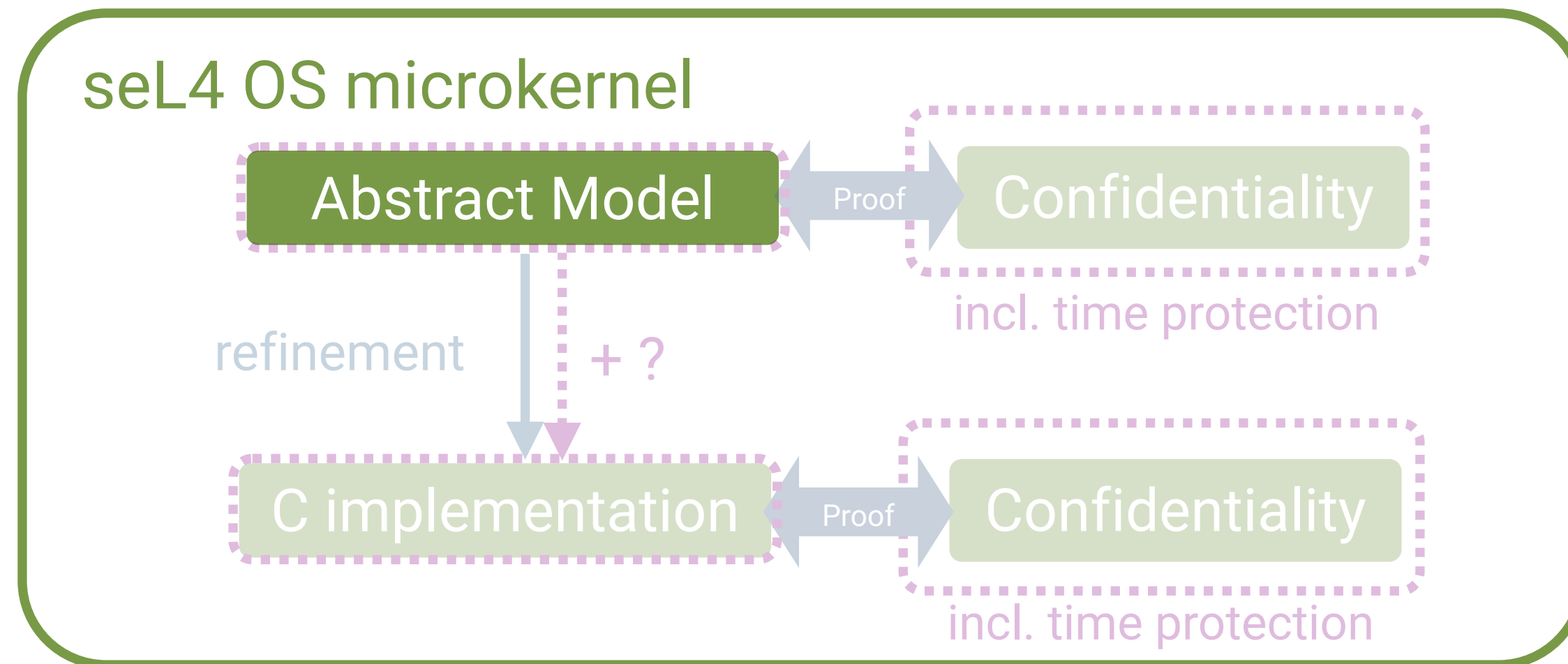


## Microkit-based OS Services

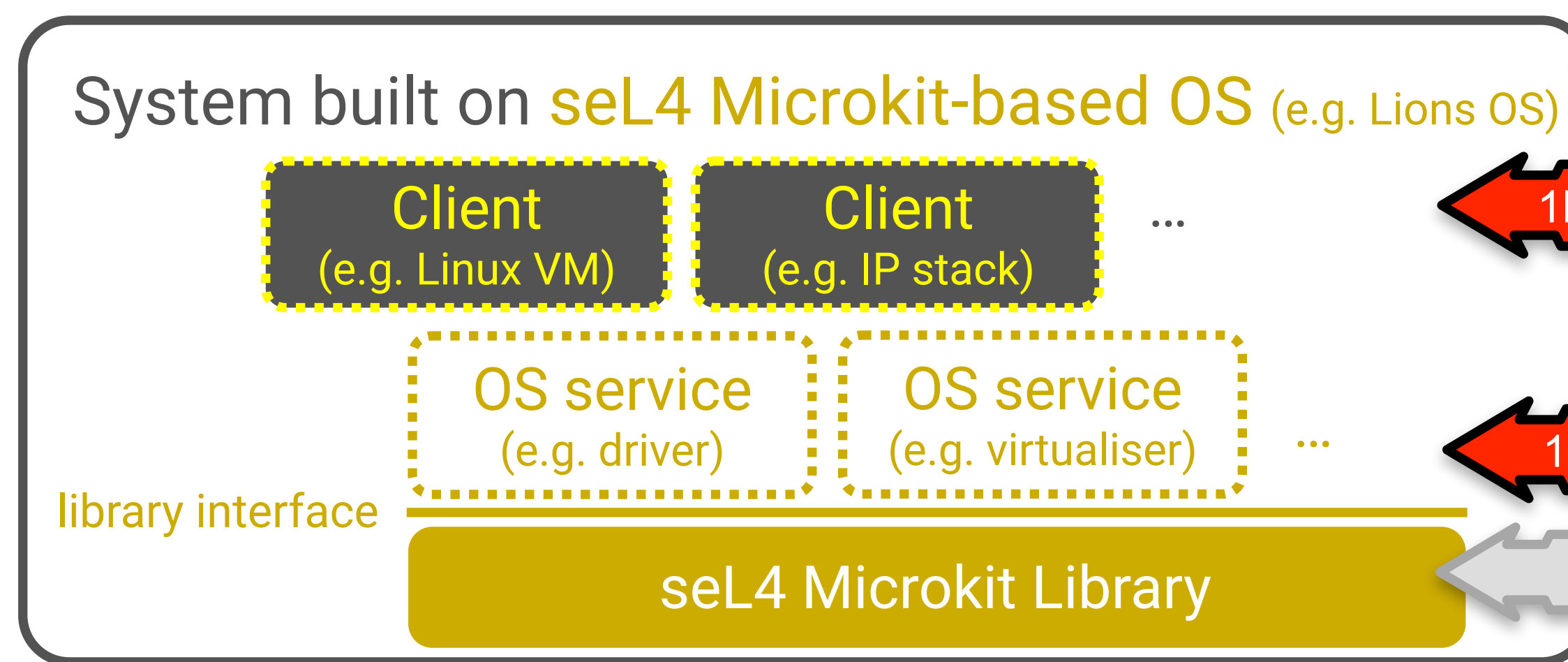


- ← 1B Verify inter-service/client communication protocols /
- ← 1D Verify devices
- ← 1C Verify service-local properties /
- ← APSys'23: Verified with SMT
- ← 1A Verify Microkit-facing spec /

## Time Protection

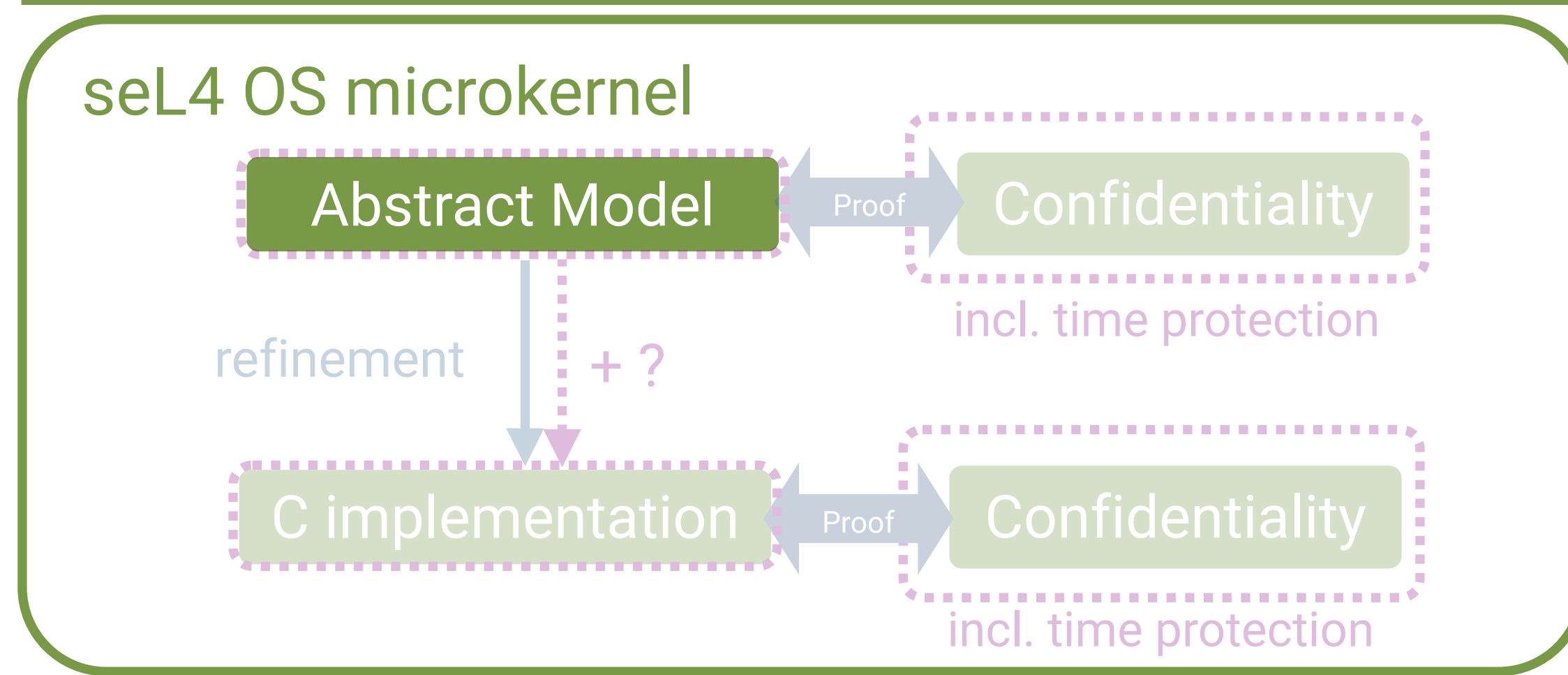


## Microkit-based OS Services



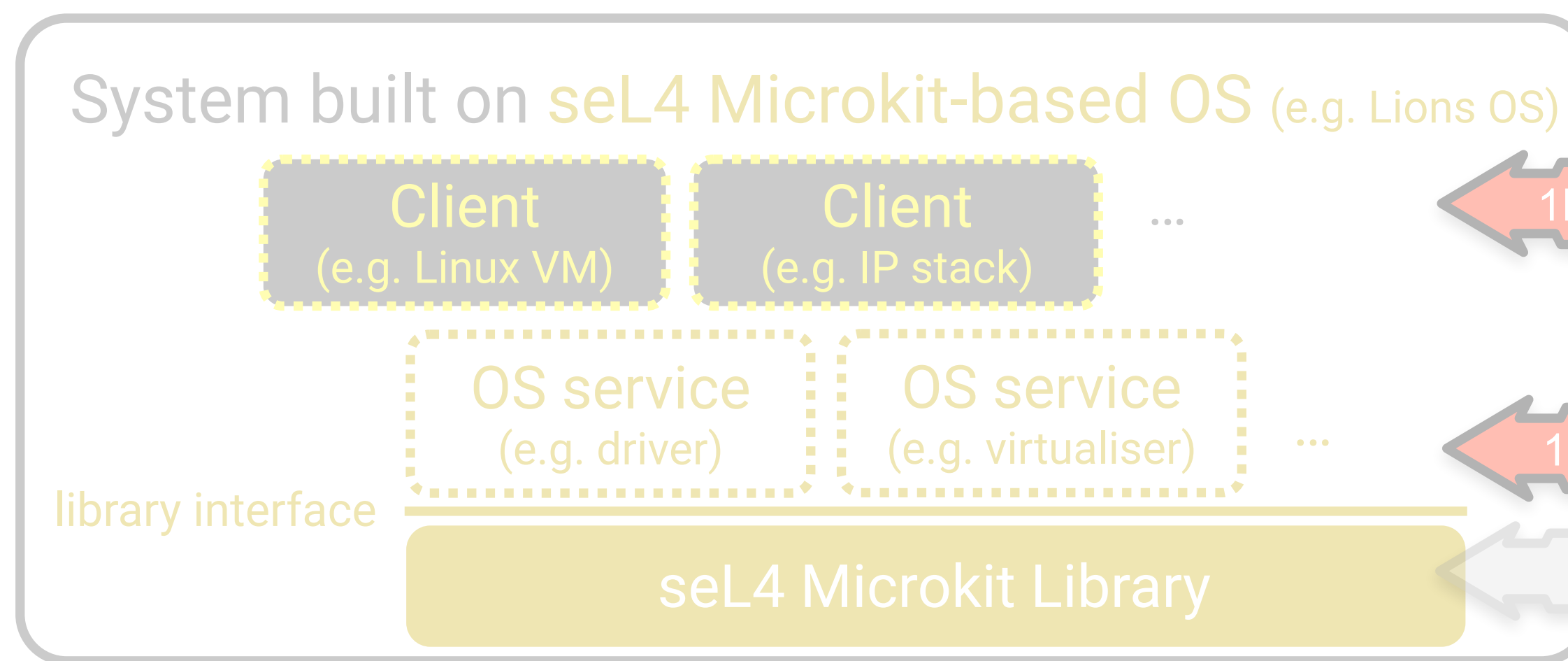
- 1E Verify system-global properties *compositionally*
- 1B Verify inter-service/client communication protocols /
- 1D Verify devices
- 1C Verify service-local properties /
- APSys'23: Verified with SMT
- 1A Verify Microkit-facing spec /

## Time Protection



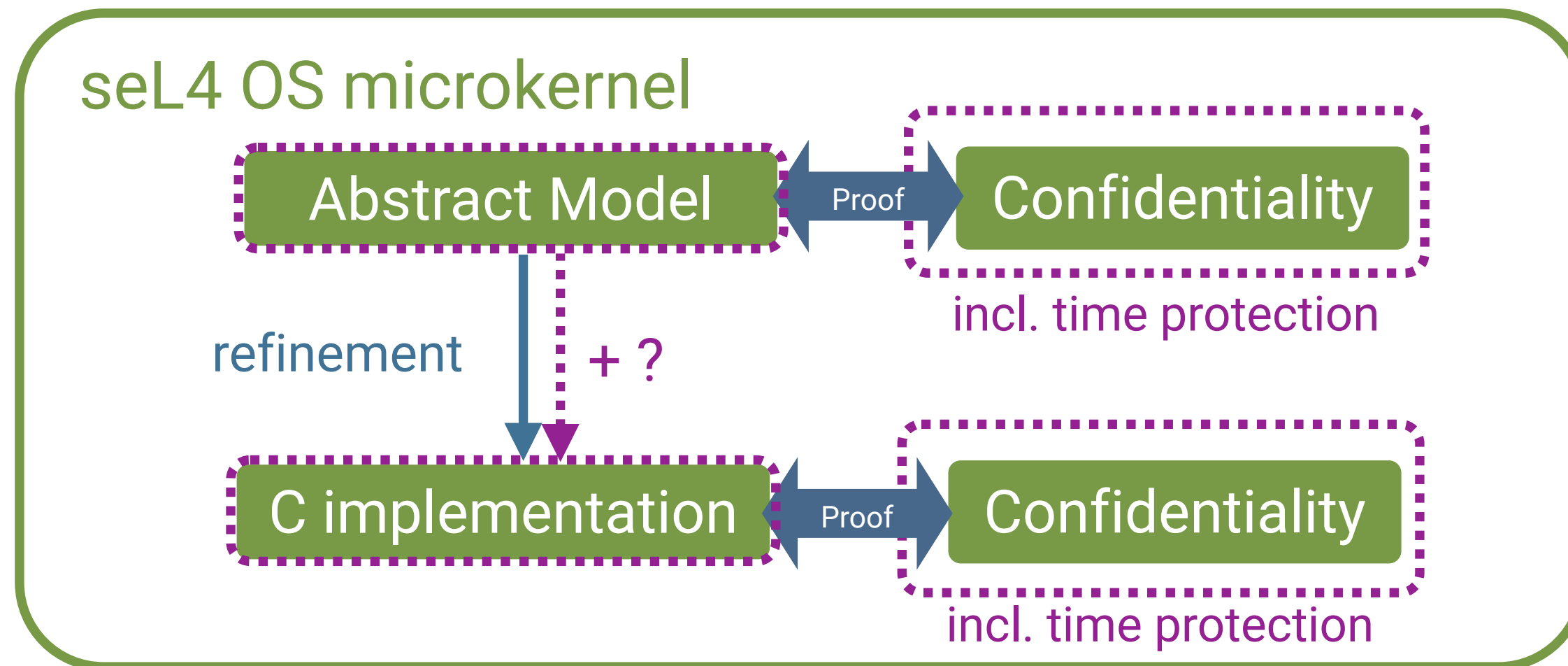
# Verification status

Microkit-based OS Services

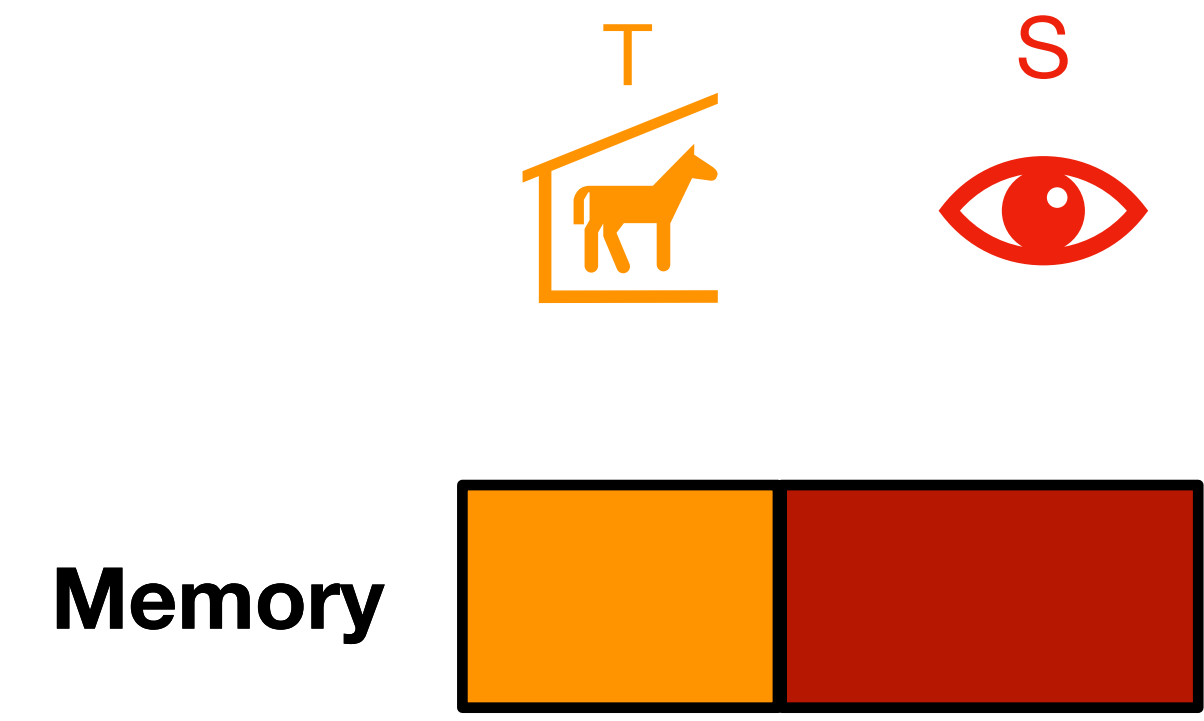


- ← 1E Verify system-global properties *compositionally*
- ← 1B Verify inter-service/client communication protocols /
- ← 1D Verify devices
- ← 1C Verify service-local properties /
- ← APSys'23: Verified with SMT
- ← 1A Verify Microkit-facing spec /

Time Protection



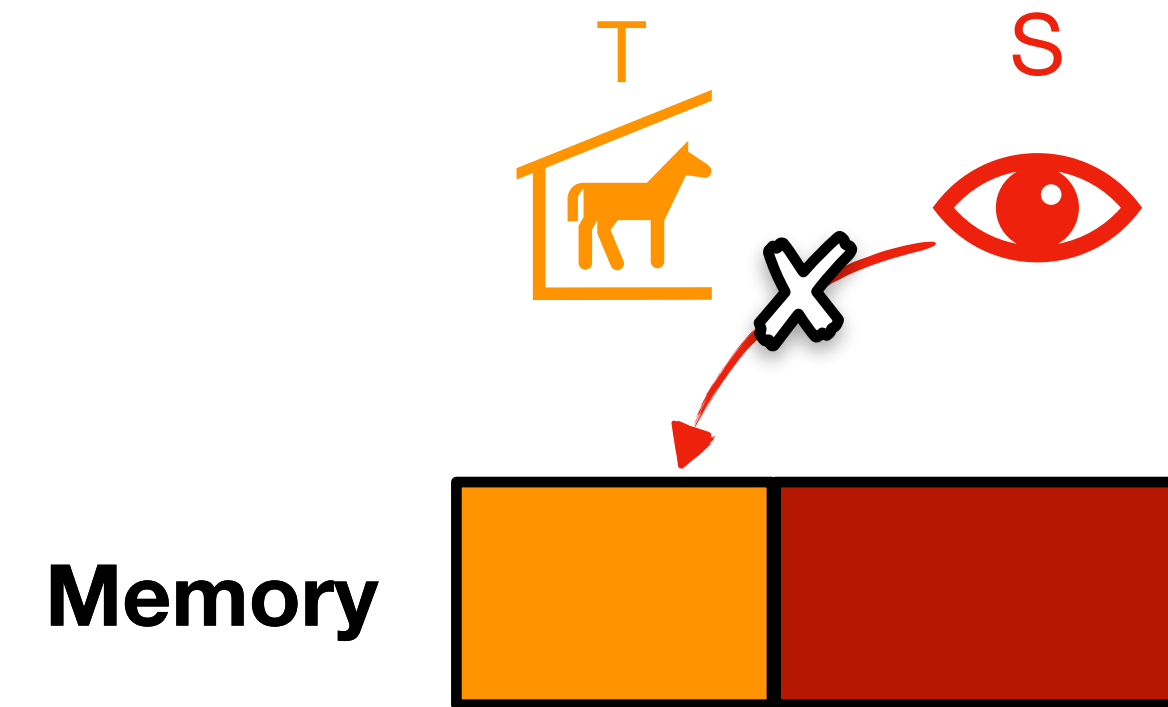
# What is Time Protection?



# What is Time Protection?



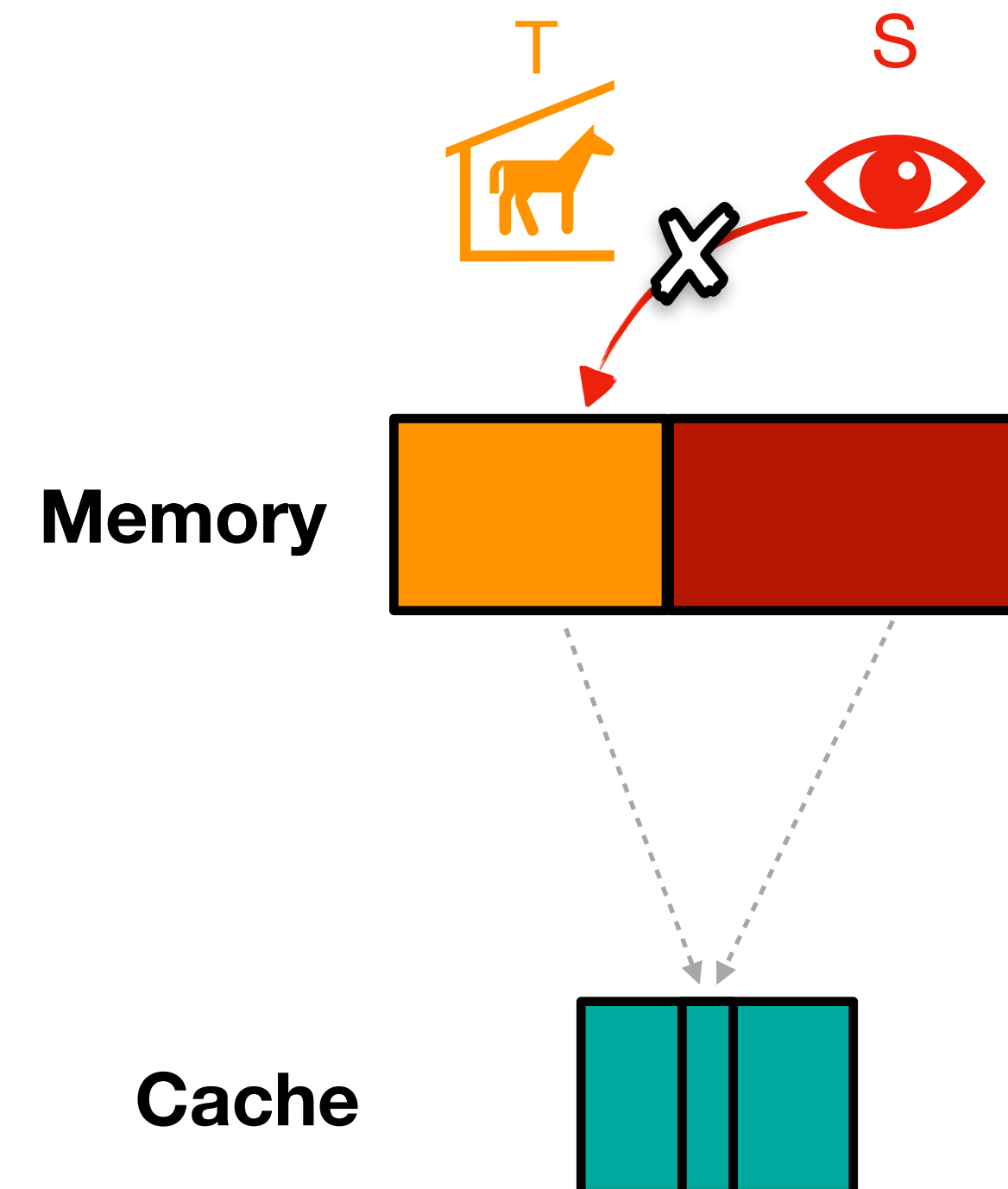
- OSes typically implement *memory protection*.



# What is Time Protection?



- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.

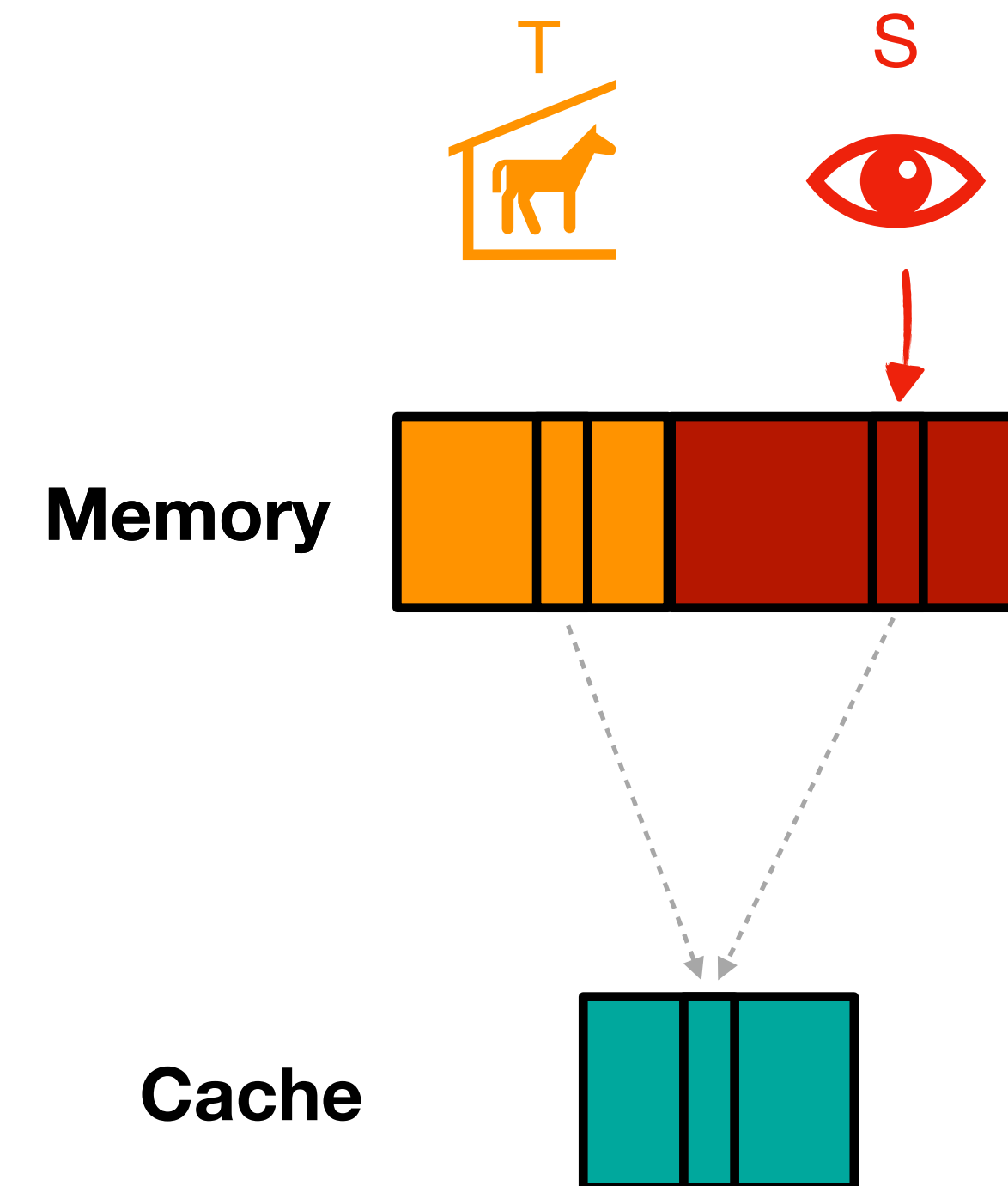




# What is Time Protection?



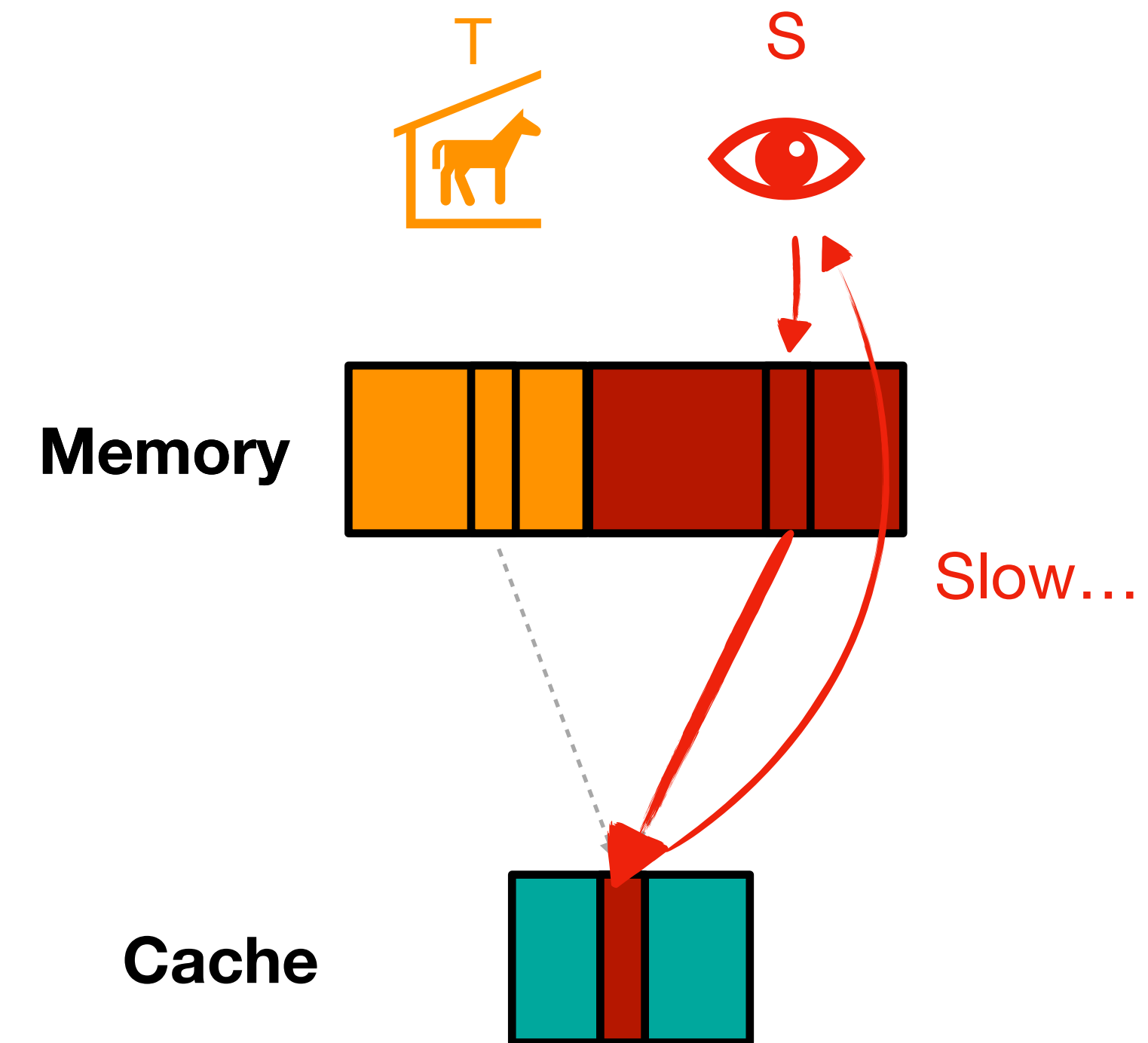
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.



# What is Time Protection?



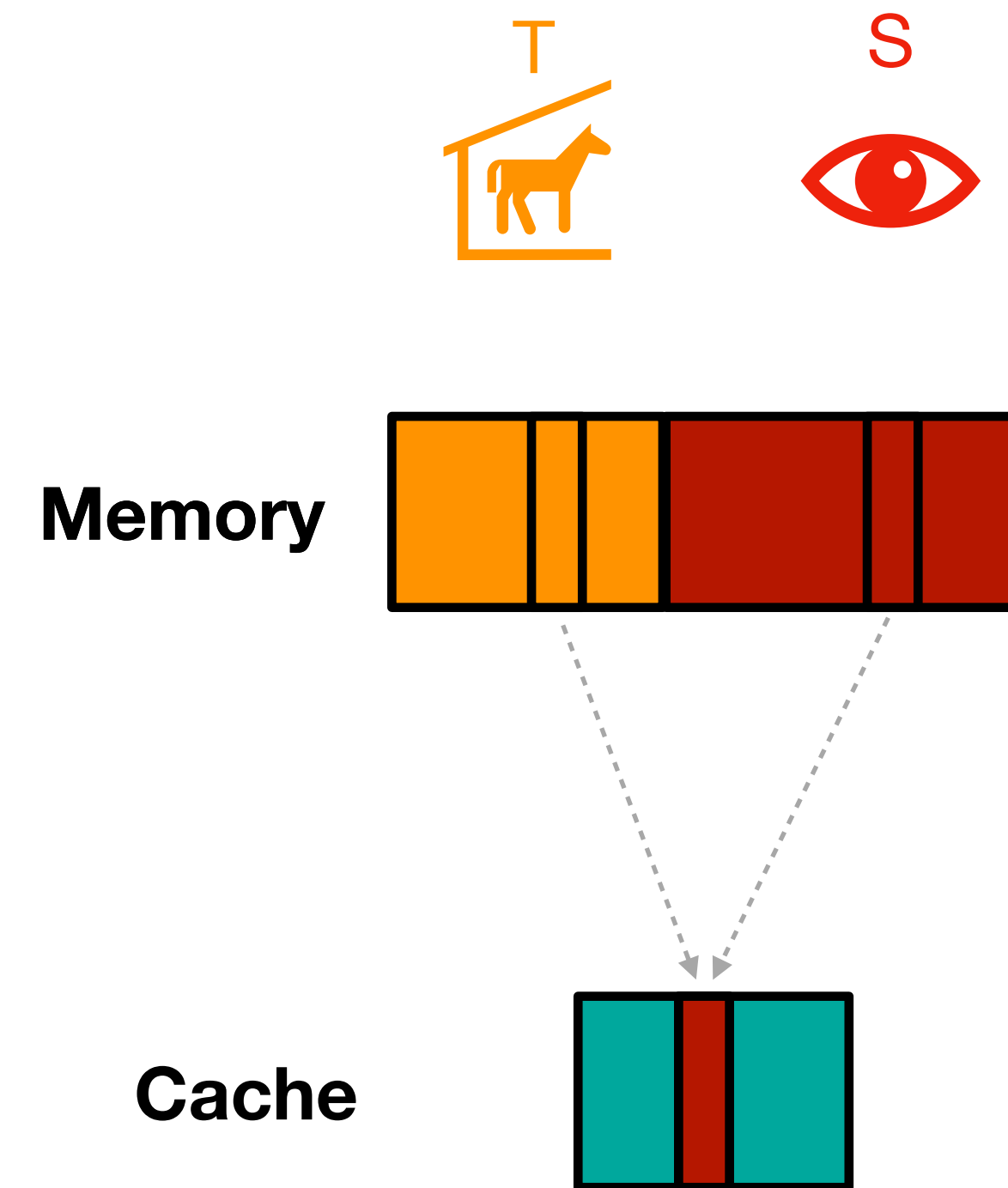
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.



# What is Time Protection?



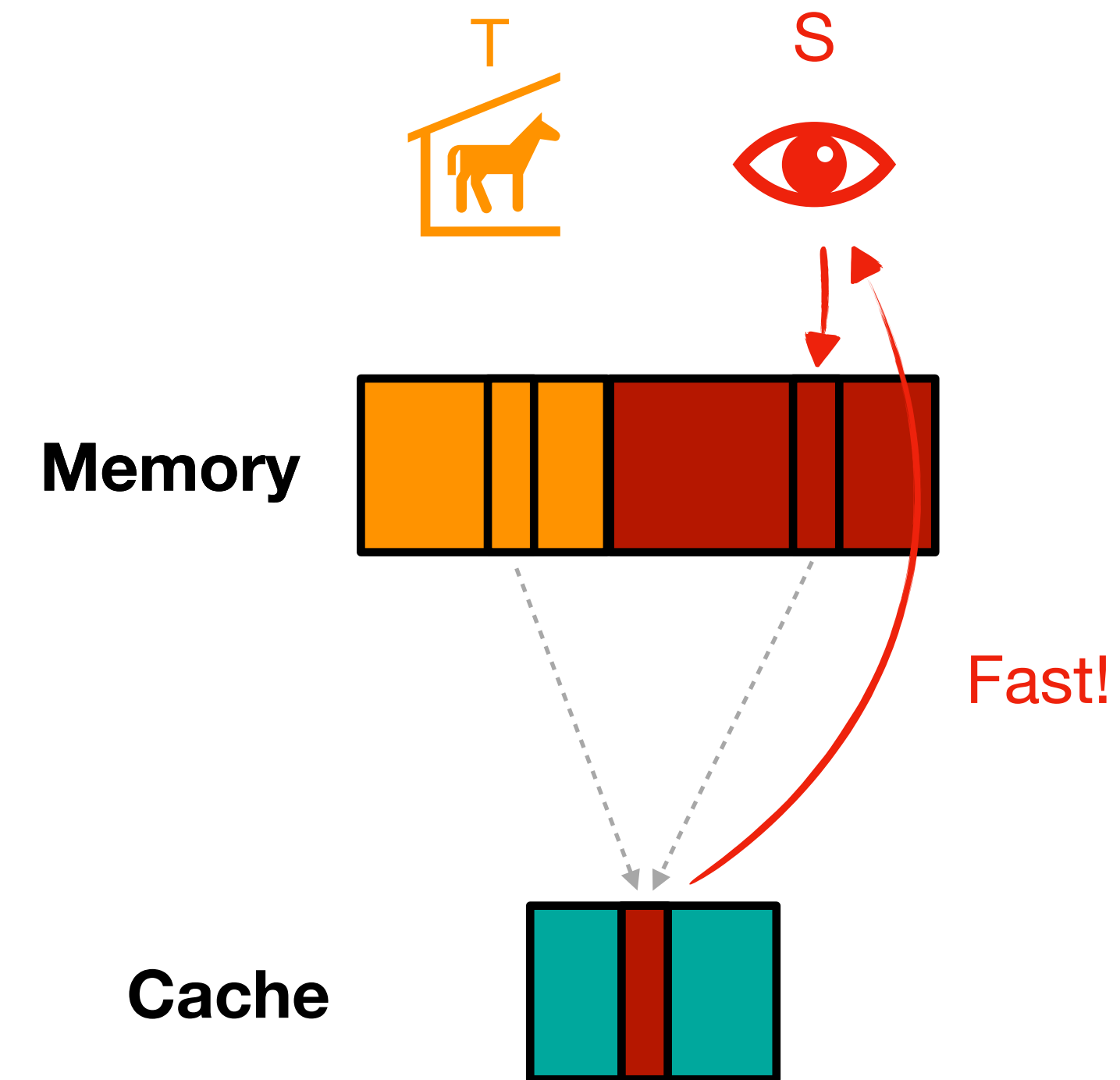
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.



# What is Time Protection?



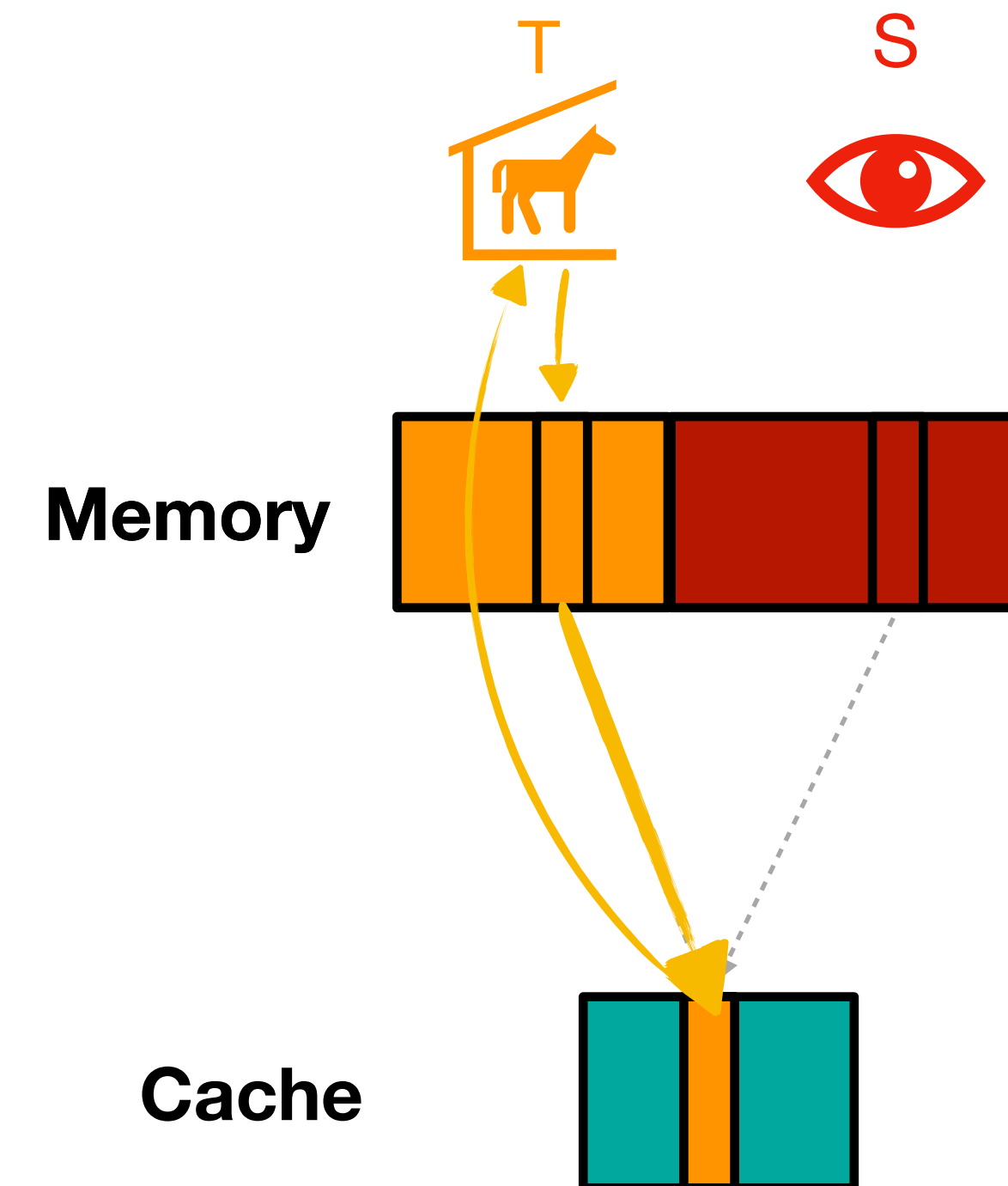
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.



# What is Time Protection?



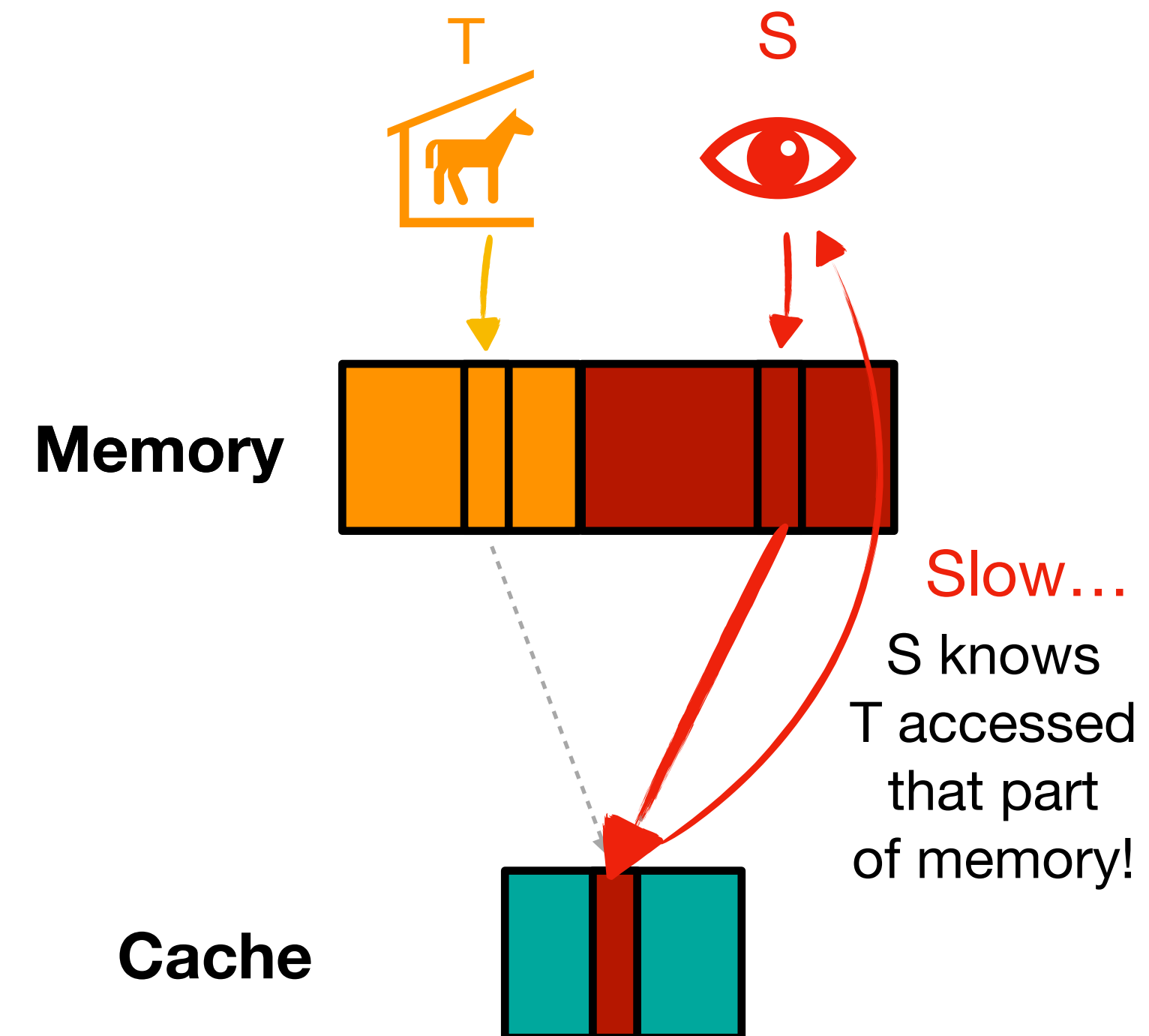
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.



# What is Time Protection?



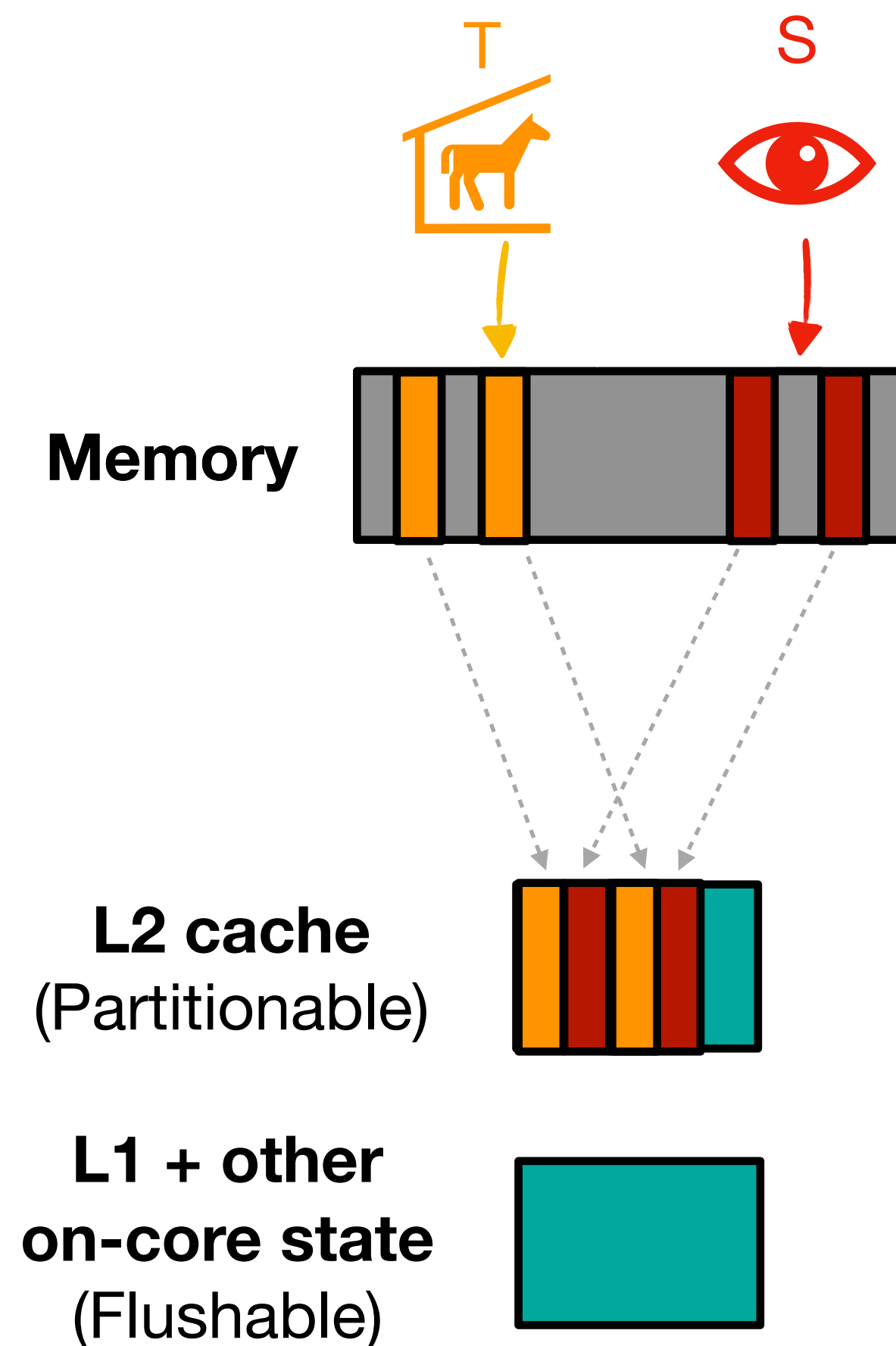
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.



# What is Time Protection?



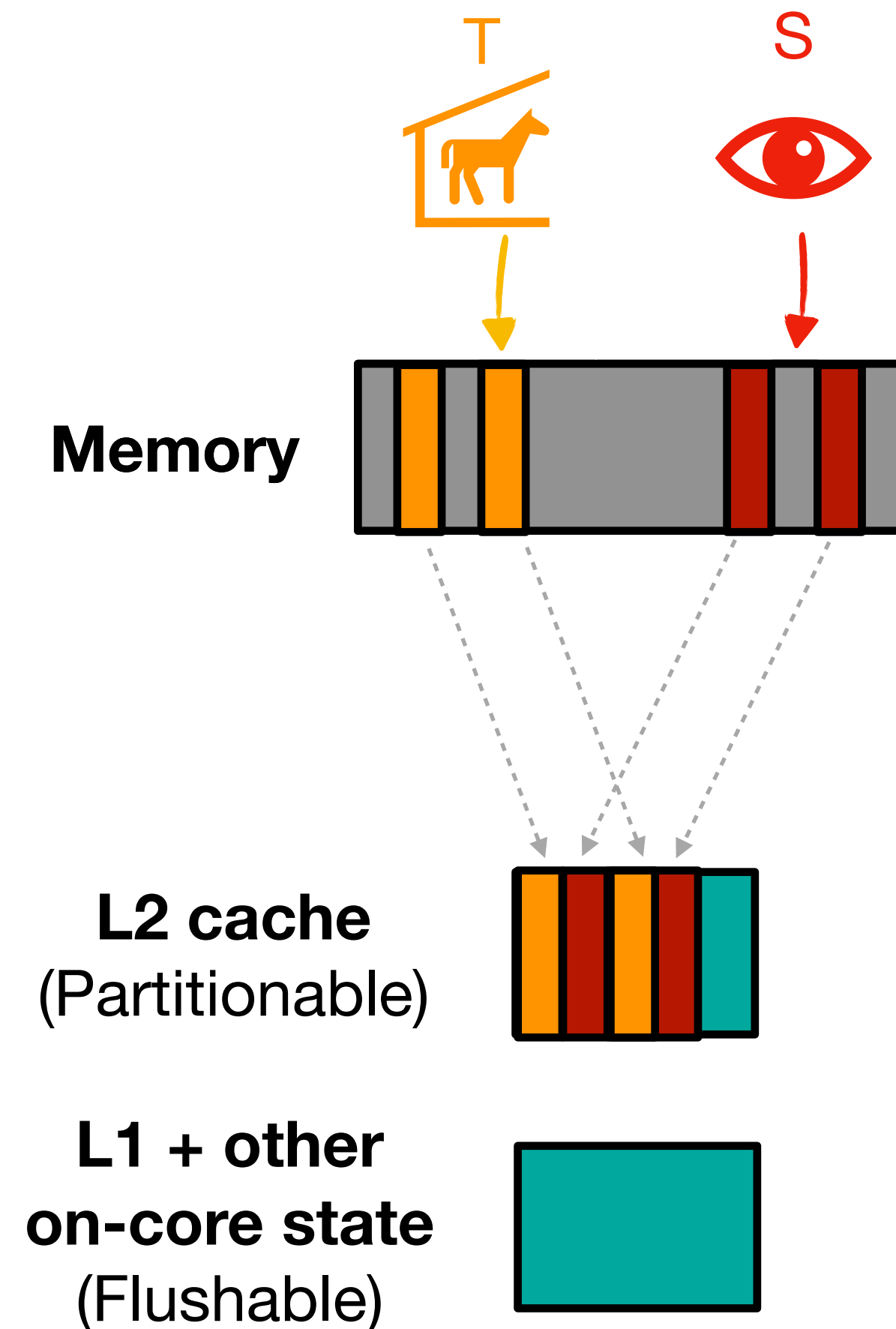
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
- *Partition* off-core memory caches



# What is Time Protection?



- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
- *Partition* off-core memory caches
- *Flush* on-core and non-architected state and *pad* time on context switch

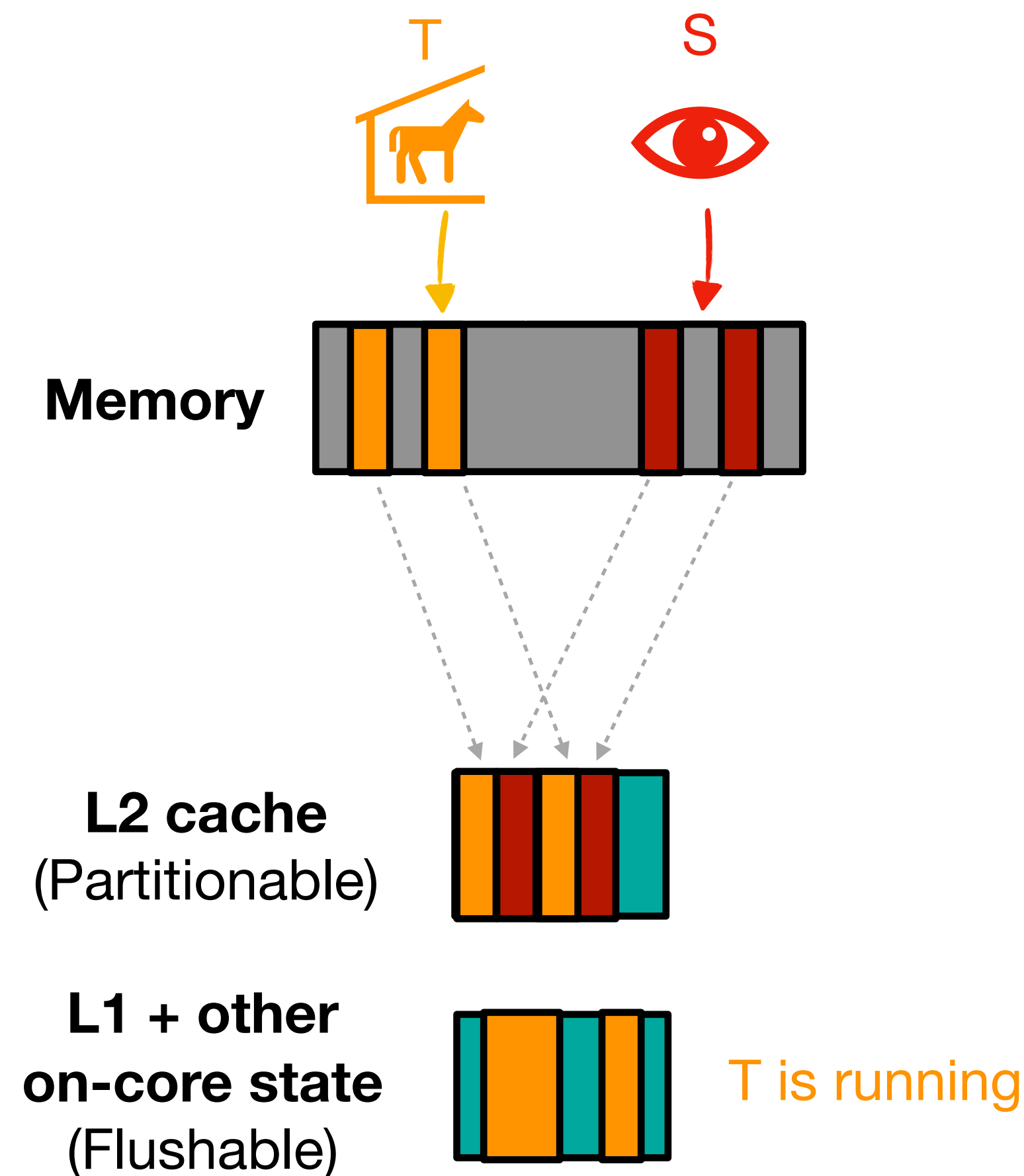




# What is Time Protection?



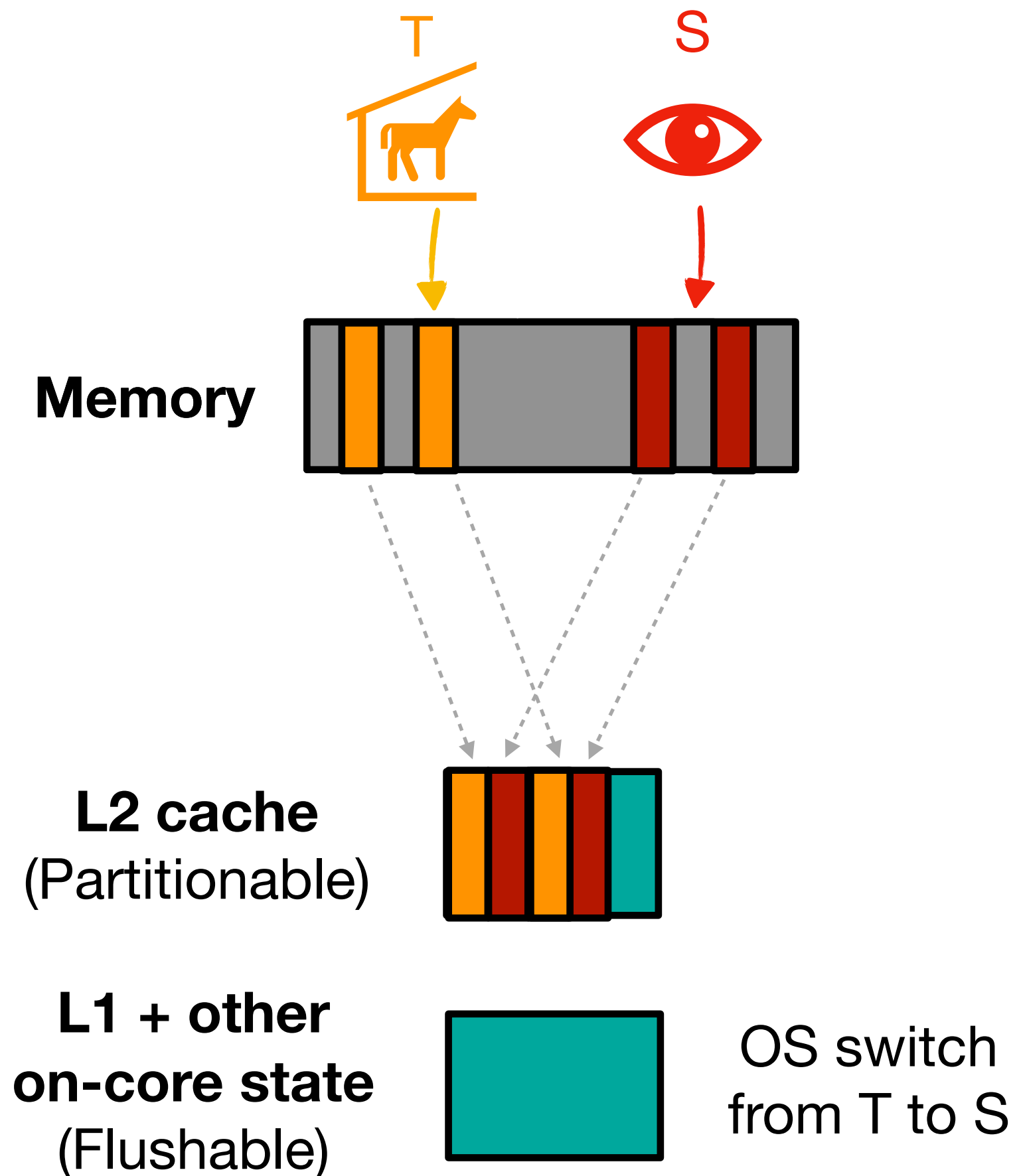
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
- *Partition* off-core memory caches
- *Flush* on-core and non-architected state and *pad* time on context switch



# What is Time Protection?



- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
- *Partition* off-core memory caches
- *Flush* on-core and non-architected state and *pad* time on context switch

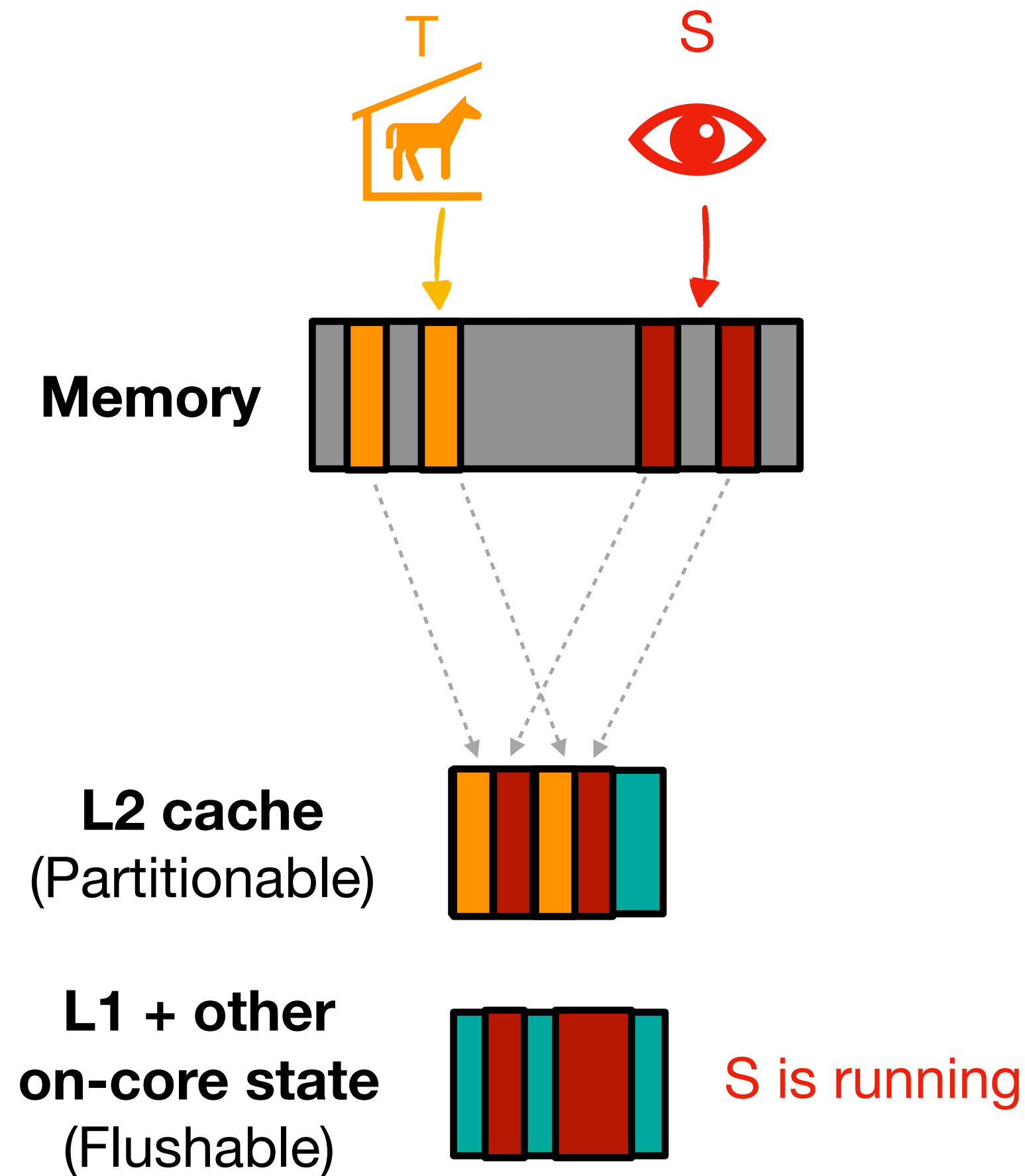


“Flush”: Write fixed content; wait up to fixed time.

# What is Time Protection?



- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
- *Partition* off-core memory caches
- *Flush* on-core and non-architected state and *pad* time on context switch

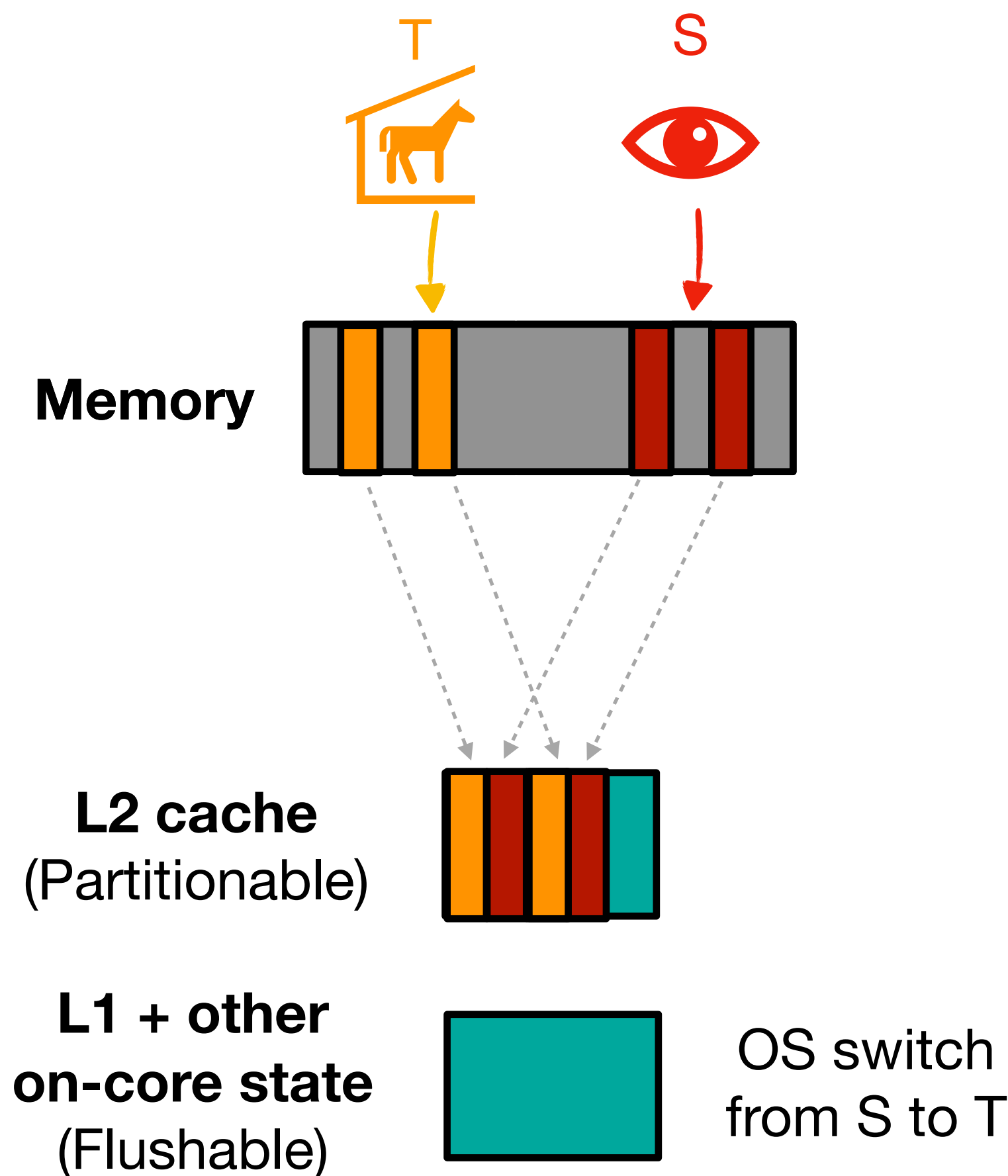


“Flush”: Write fixed content; wait up to fixed time.

# What is Time Protection?



- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
- *Partition* off-core memory caches
- *Flush* on-core and non-architected state and *pad* time on context switch

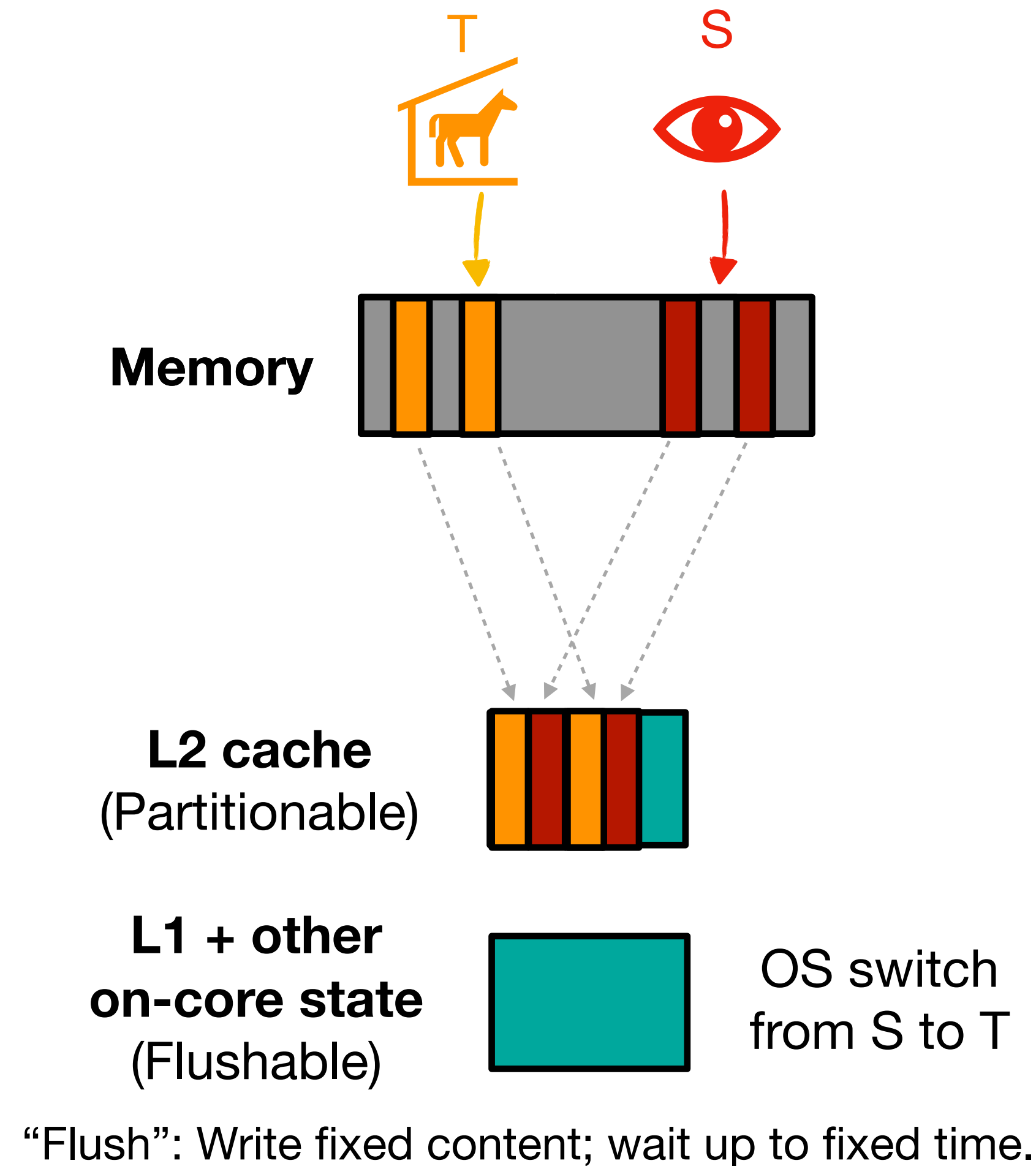


“Flush”: Write fixed content; wait up to fixed time.

# What is Time Protection?



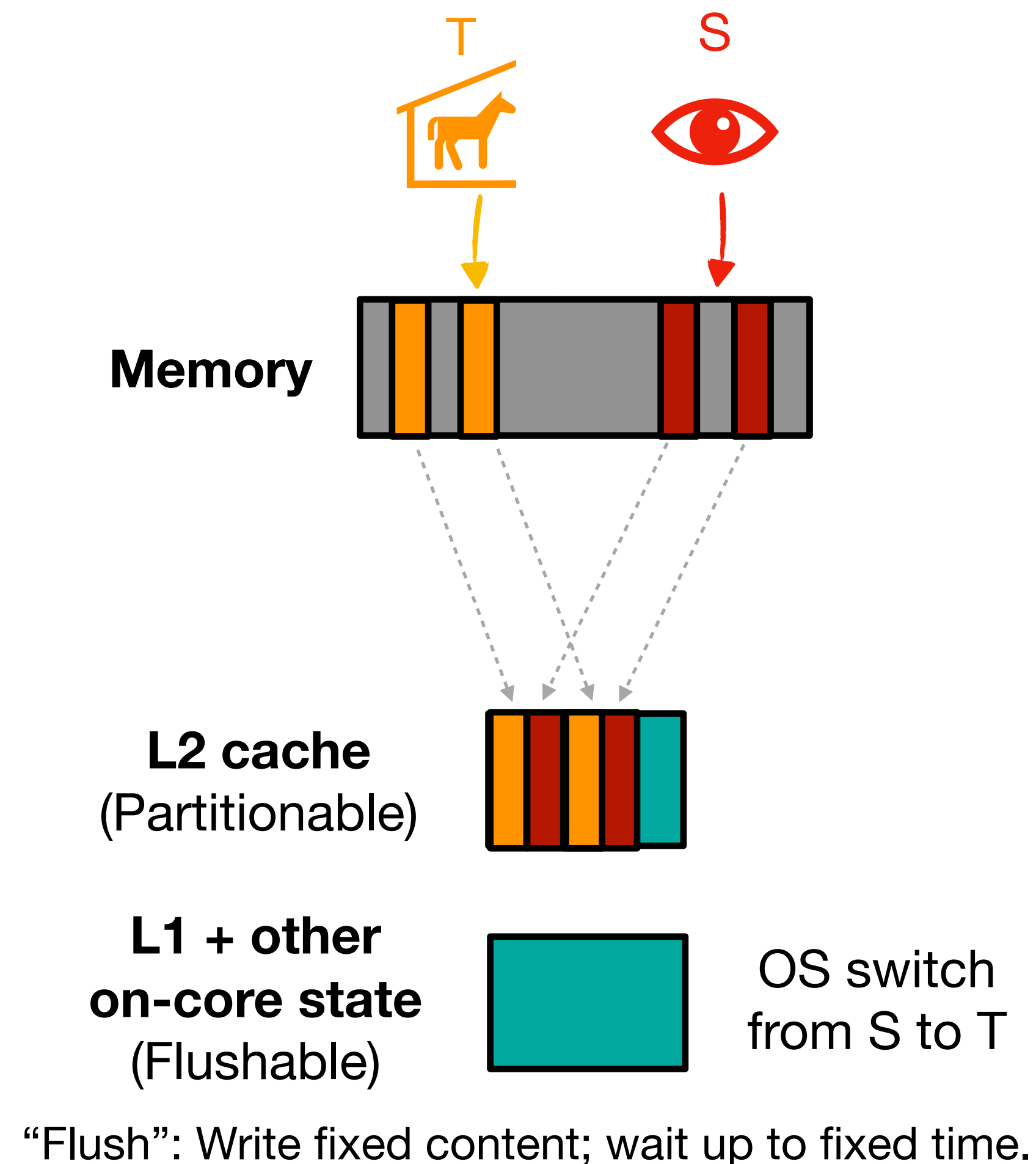
- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
  - *Partition* off-core memory caches
  - *Flush* on-core and non-architected state and *pad* time on context switch
- seL4 OS kernel's enforcement of time protection:
  - Implemented, evaluated empirically on ARM, x86  
See EuroSys: [Ge et al. 2019]



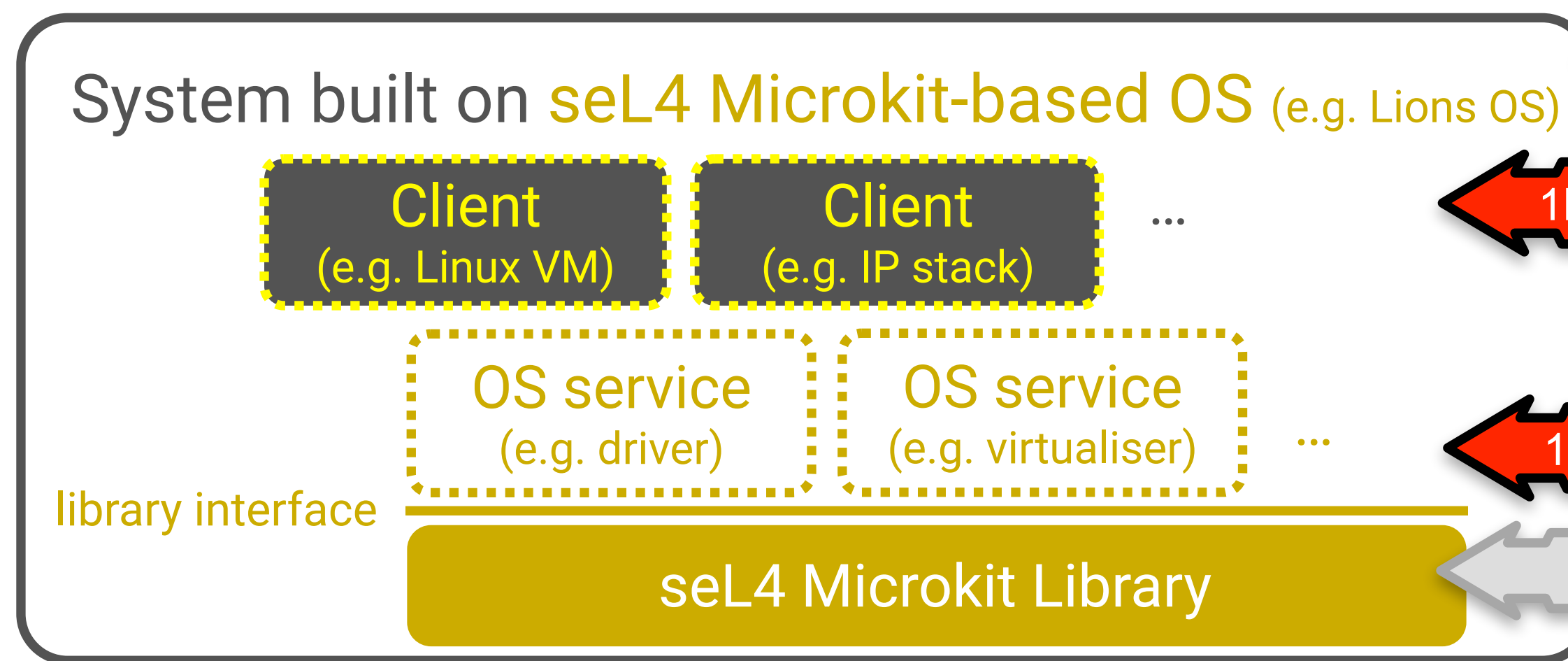
# What is Time Protection?



- OSes typically implement *memory protection*.
- But: Mere memory access changes microarchitectural state — this affects timing.
- To prevent these *timing channels*, OSes can implement *time protection*:  
See EuroSys: [Ge et al. 2019]
  - *Partition* off-core memory caches
  - *Flush* on-core and non-architected state and *pad* time on context switch
- seL4 OS kernel's enforcement of time protection:
  - Implemented, evaluated empirically on ARM, x86  
See EuroSys: [Ge et al. 2019]
  - Ported to RISC-V with hardware support  
See arXiv preprint: [Buckley, Sison et al. 2023]  
HW support: [Wistoff et al. 2023]

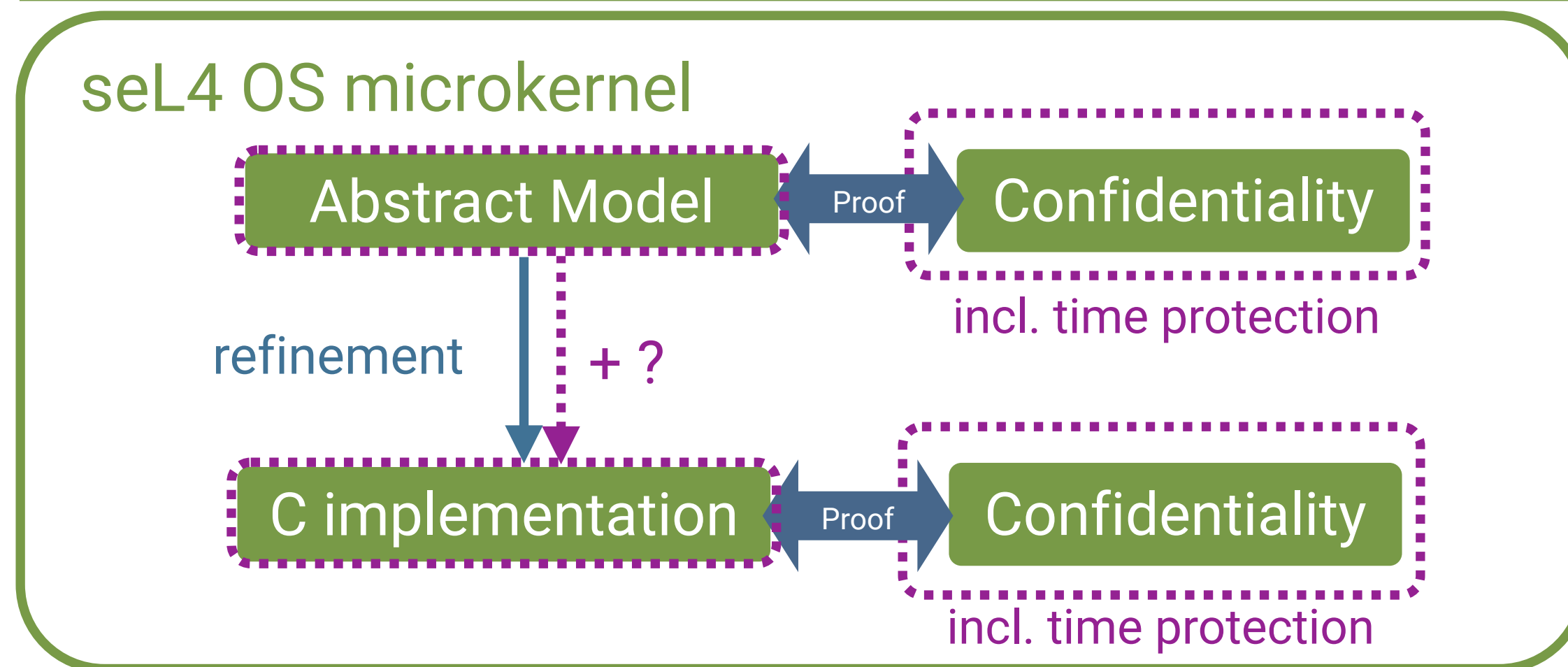


## Microkit-based OS Services



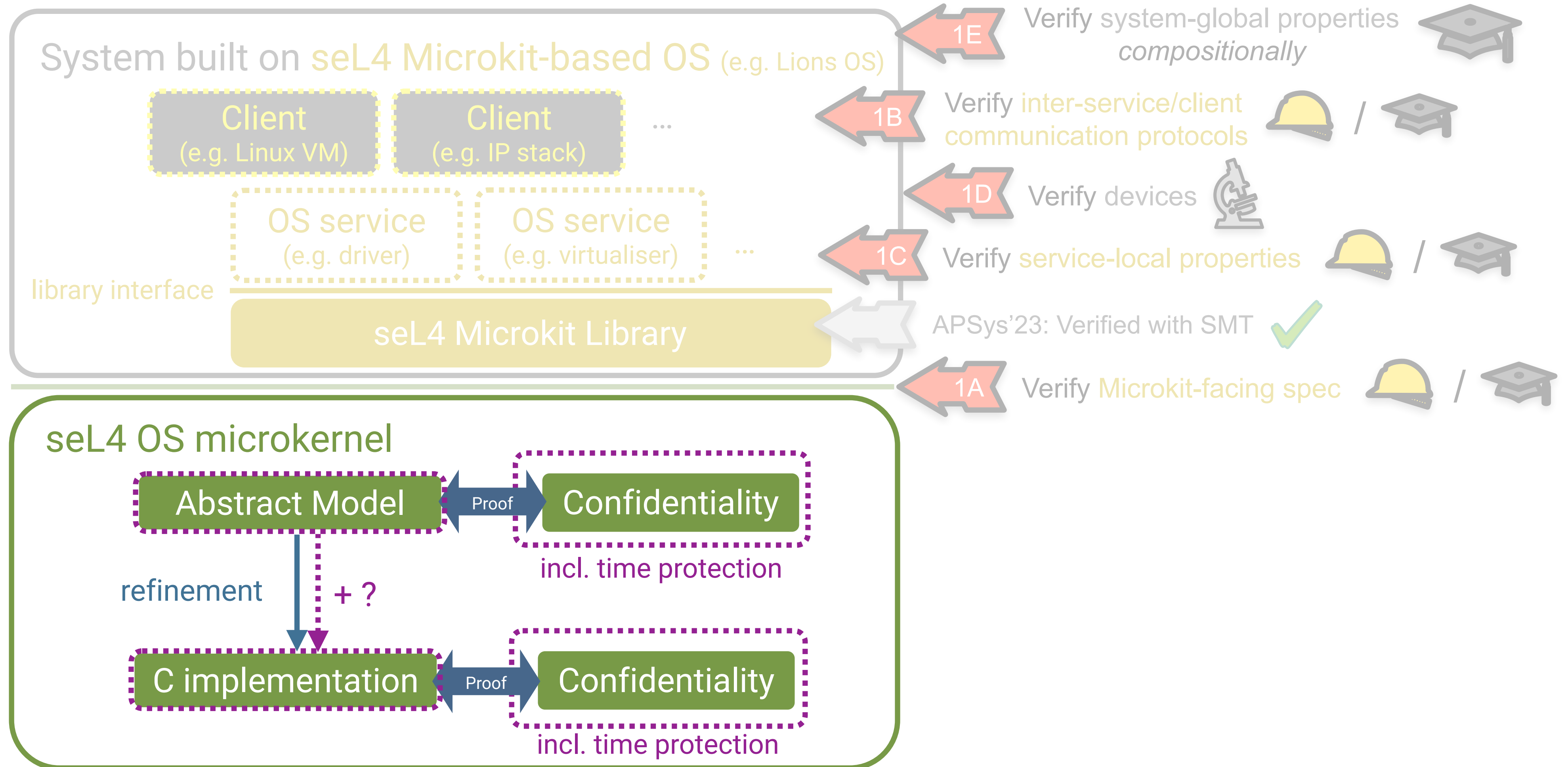
- 1E Verify system-global properties *compositionally*
- 1B Verify inter-service/client communication protocols /
- 1D Verify devices
- 1C Verify service-local properties /
- APSys'23: Verified with SMT
- 1A Verify Microkit-facing spec /

## Time Protection



# Verification status

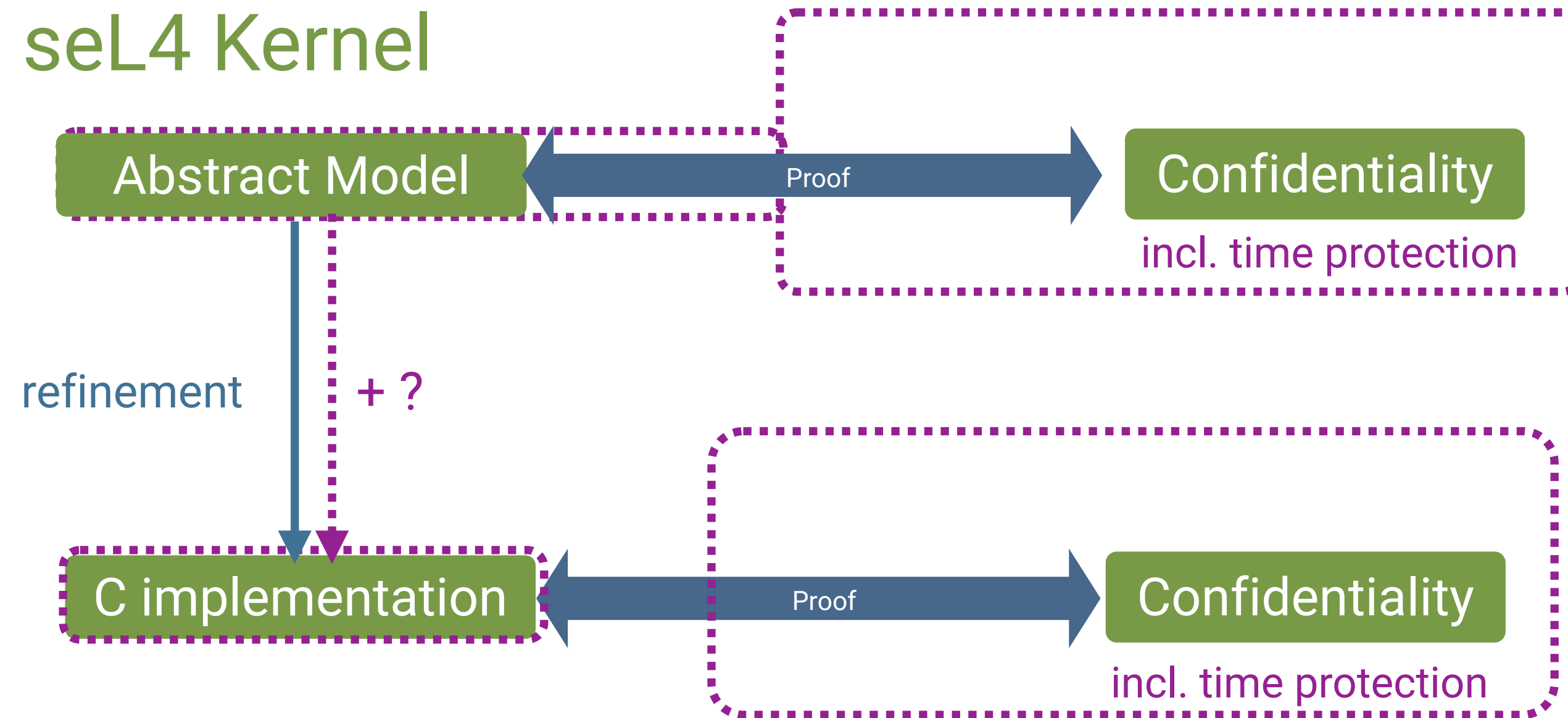
Microkit-based OS Services



Time Protection



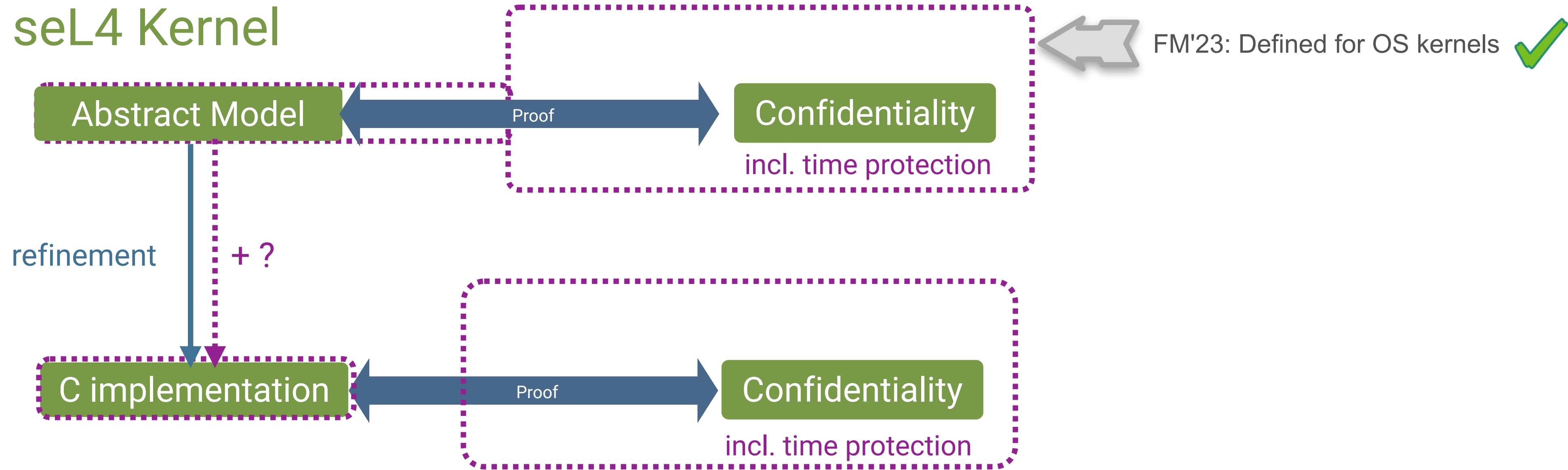
# Proving seL4 implements Time Protection



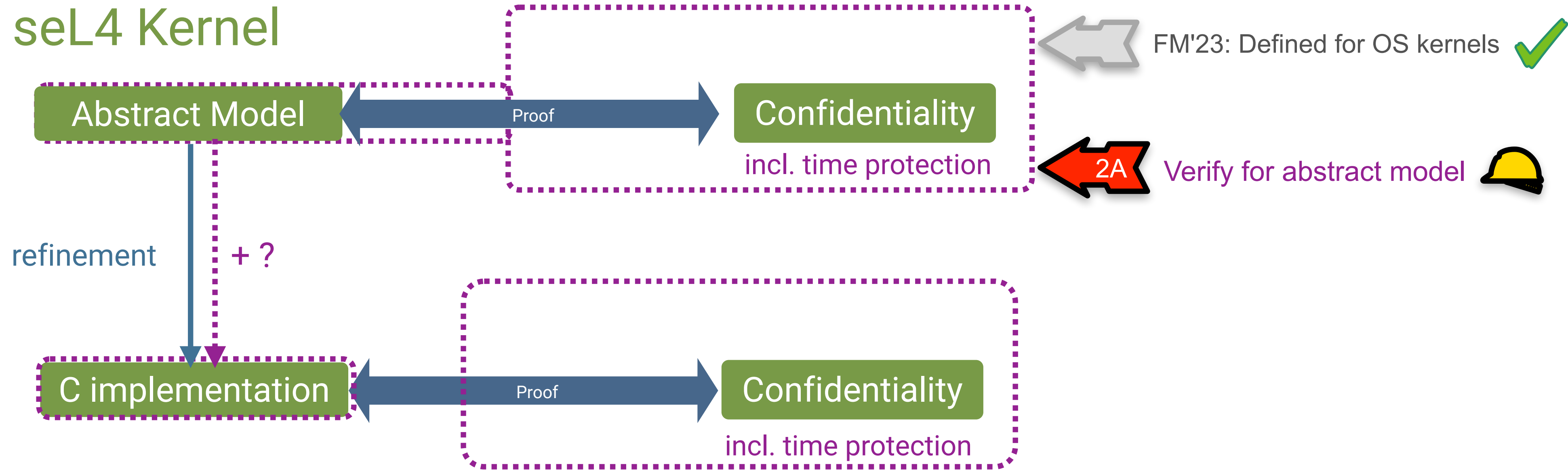
# Proving seL4 implements Time Protection



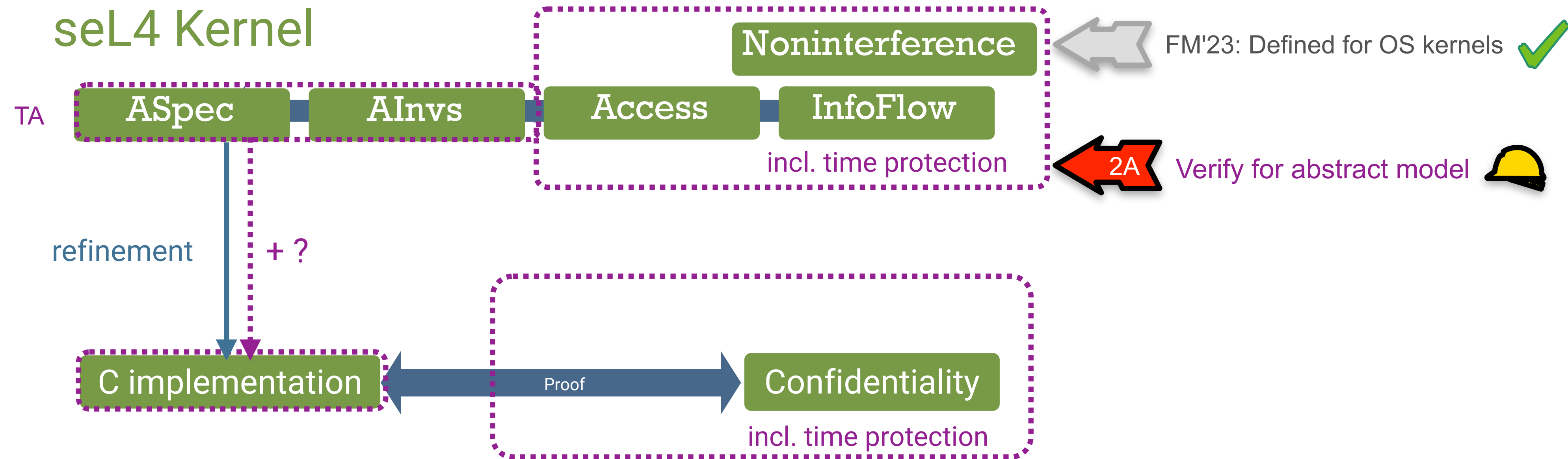
seL4 Kernel



# Proving seL4 implements Time Protection

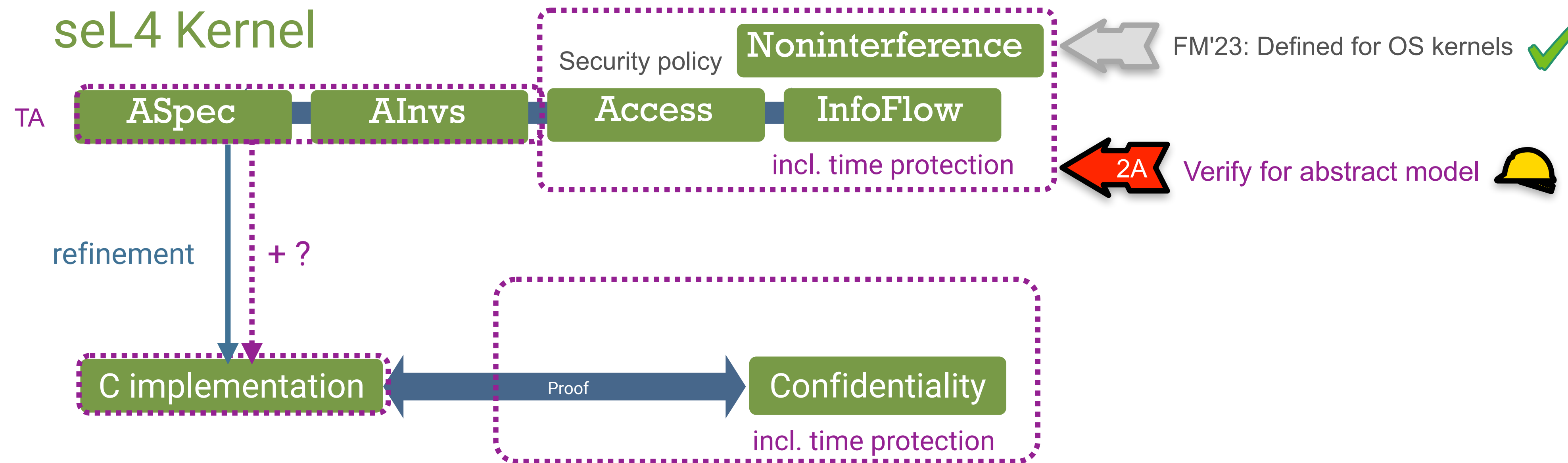


# Proving seL4 implements Time Protection



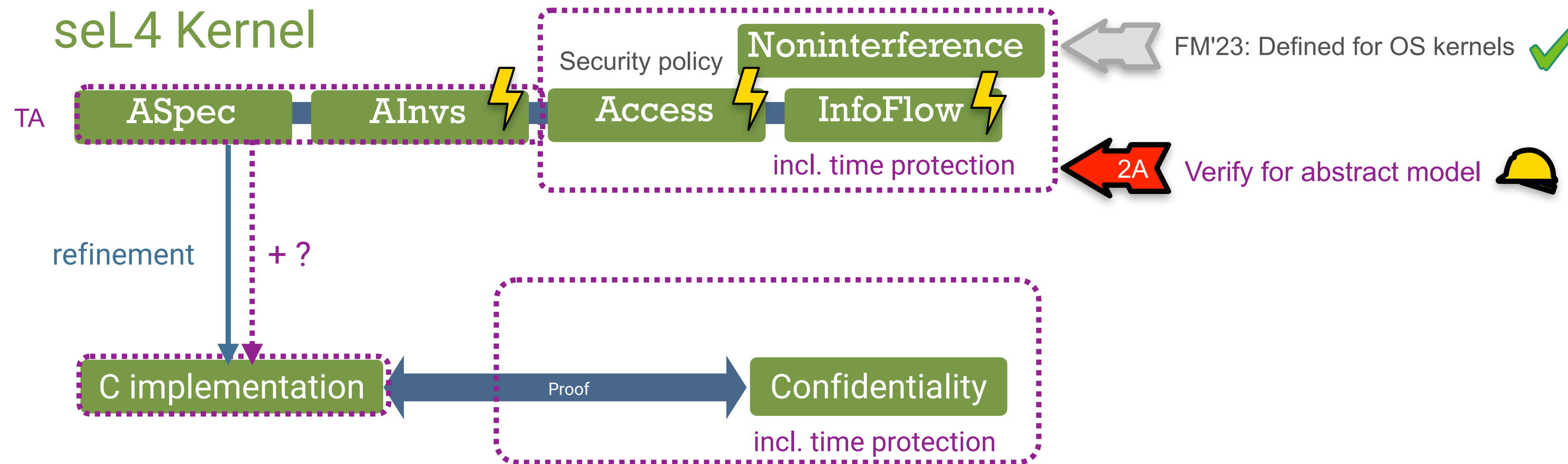
- Abstract model (**ASpec**) checks *touched addresses* (TA) set

# Proving seL4 implements Time Protection



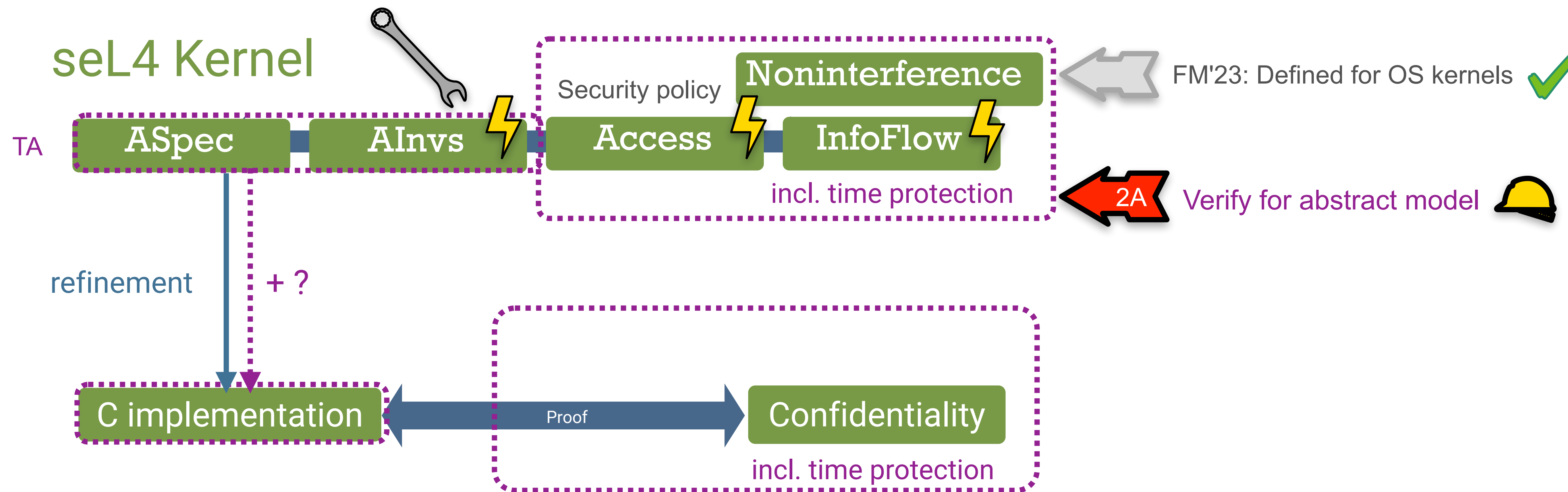
- Abstract model (**ASpec**) checks *touched addresses* (TA) set
- (2A) Key **ASpec** property:  $TA \subseteq \text{domain's addresses}$  (according to *security policy*)

# Proving seL4 implements Time Protection



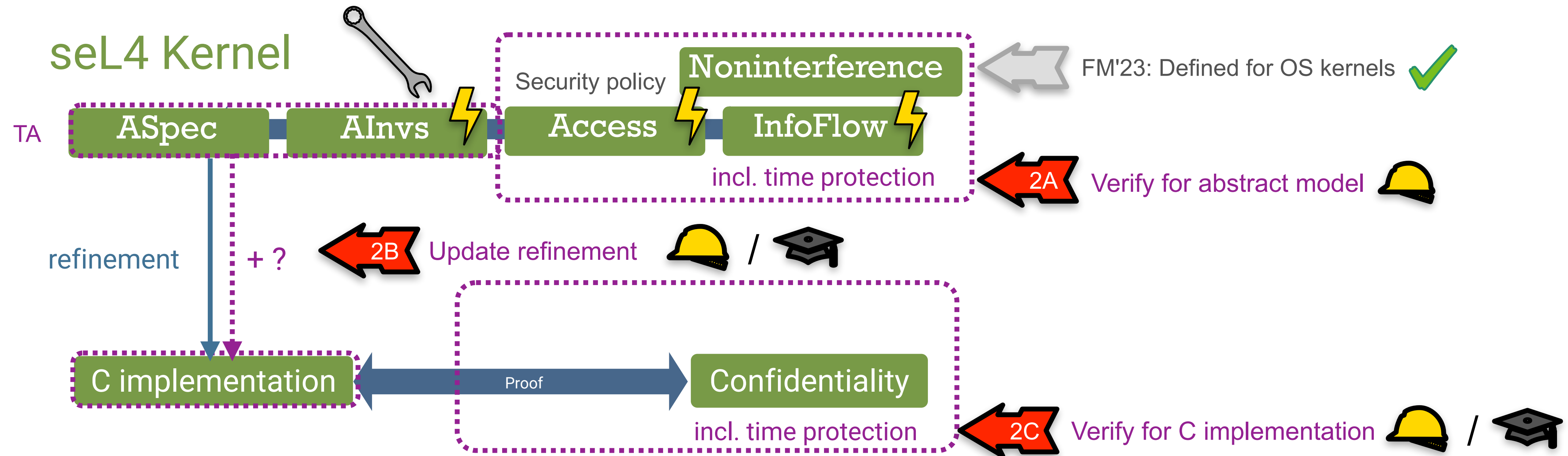
- Abstract model (**ASpec**) checks *touched addresses* (TA) set
- (2A) Key **ASpec** property:  $TA \subseteq \text{domain's addresses}$  (according to *security policy*)
- Also: **AInvs**, **Access**, **InfoFlow** need repairs  

# Proving seL4 implements Time Protection



- Abstract model (**ASpec**) checks *touched addresses* (TA) set
- (2A) Key **ASpec** property:  $TA \subseteq \text{domain's addresses}$  (according to *security policy*)
- Also: **AInvs**, **Access**, **InfoFlow** need repairs  

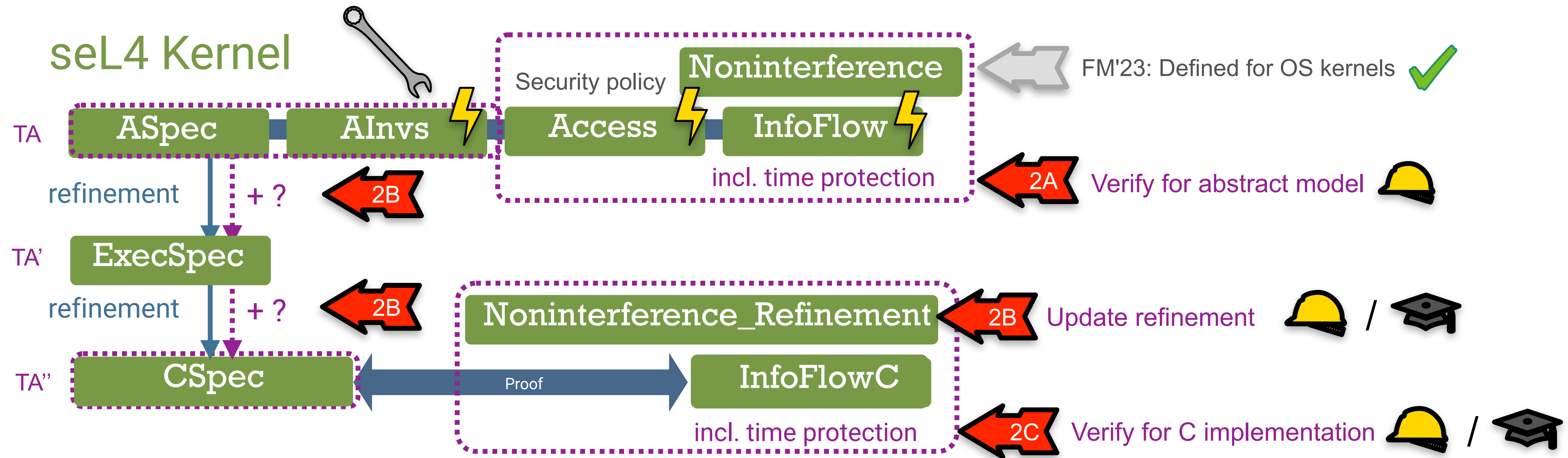
# Proving seL4 implements Time Protection



- Abstract model (**ASpec**) checks *touched addresses* (TA) set
- (2A) Key **ASpec** property:  $TA \subseteq \text{domain's addresses}$  (according to *security policy*)
- Also: **AInvs**, **Access**, **InfoFlow** need repairs  

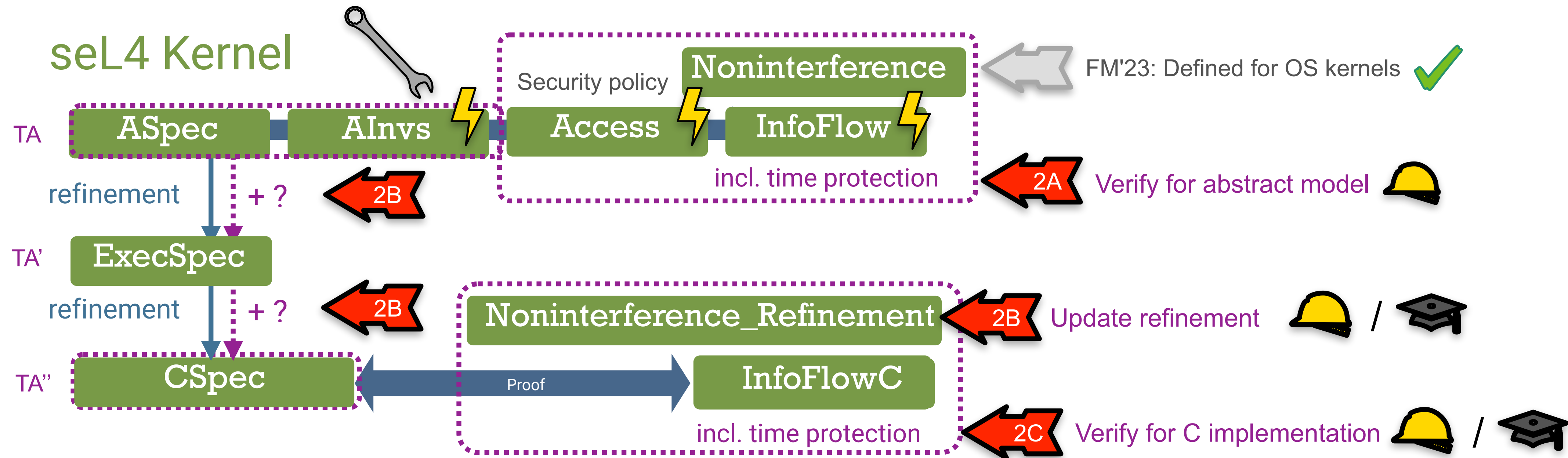


# Proving seL4 implements Time Protection



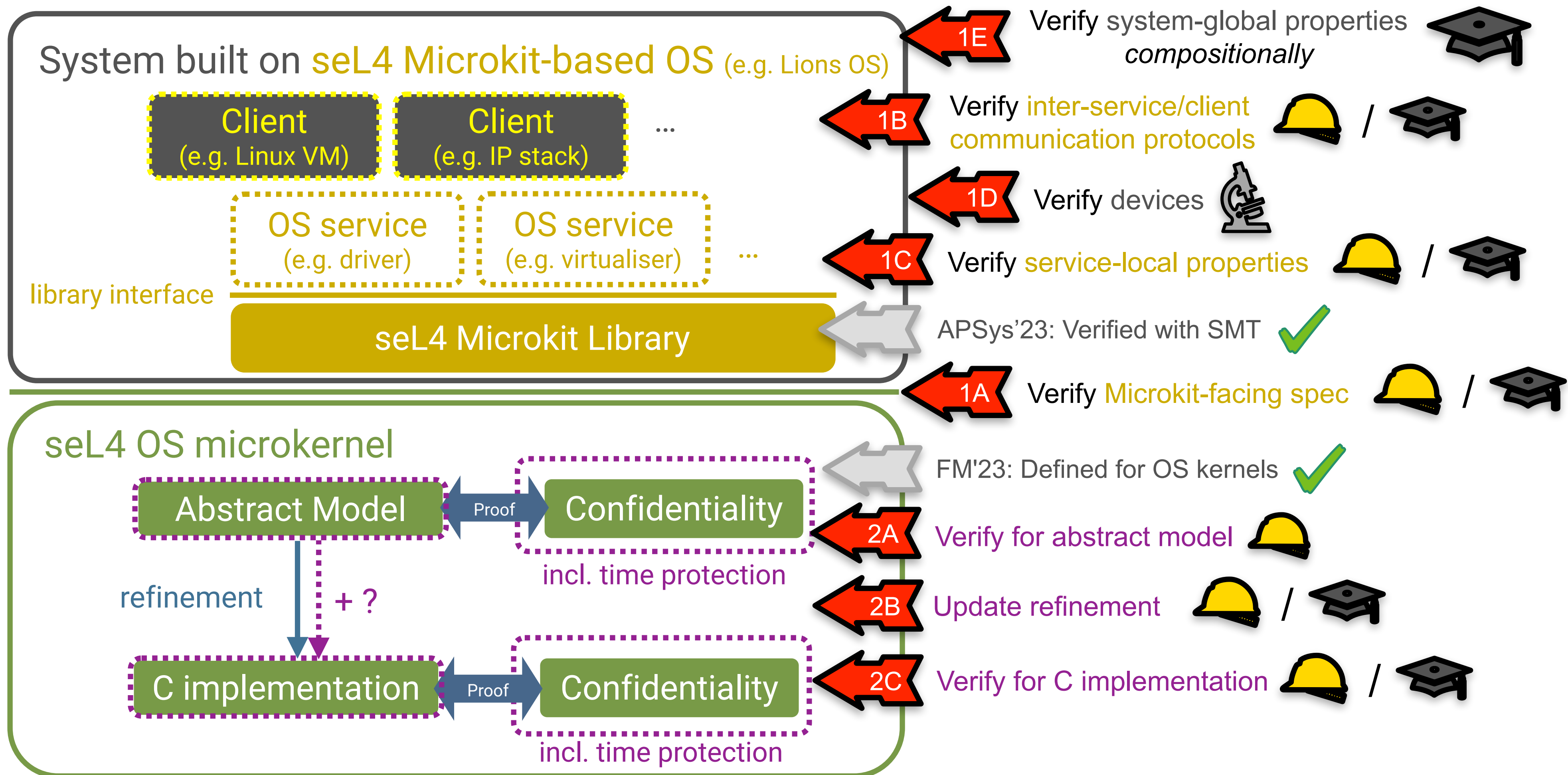
- Abstract model (**ASpec**) checks *touched addresses* (TA) set
- (2A) Key **ASpec** property:  $TA \subseteq \text{domain's addresses}$  (according to *security policy*)
  - Also: **AInvs**, **Access**, **InfoFlow** need repairs
- (2B + 2C) Refinement:  $TA'' \subseteq TA' \subseteq TA$  ? (via **ExecSpec**)

# Proving seL4 implements Time Protection



- Abstract model (**ASpec**) checks *touched addresses* (TA) set
- (2A) Key **ASpec** property:  $TA \subseteq \text{domain's addresses}$  (according to *security policy*)
  - Also: **AInvs**, **Access**, **InfoFlow** need repairs
- (2B + 2C) Refinement:  $TA'' \subseteq TA' \subseteq TA$  ? (via **ExecSpec**)
  - Modulo: Detail on addresses added by refinement to **CSpec**

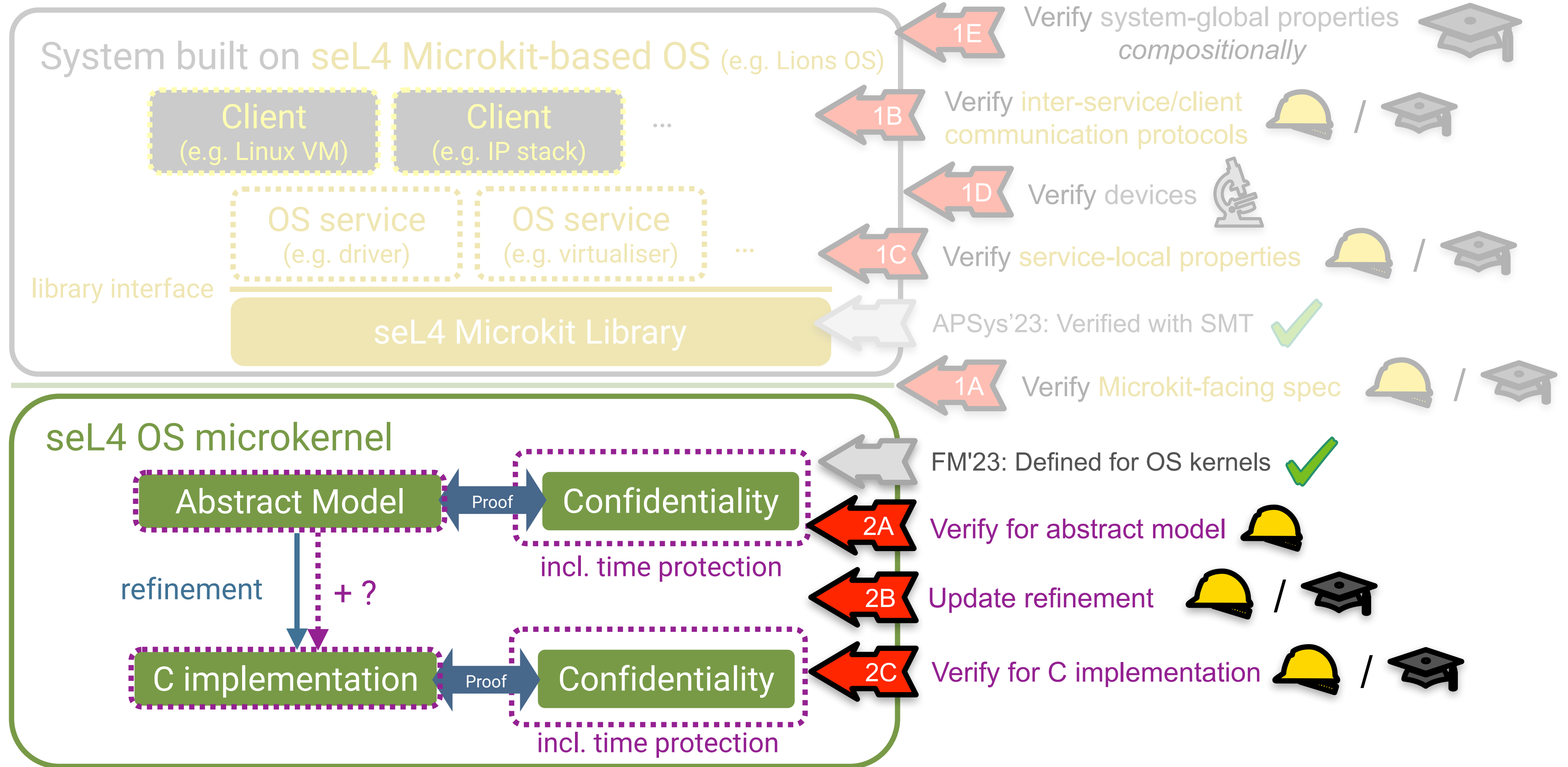
## Microkit-based OS Services



## Time Protection

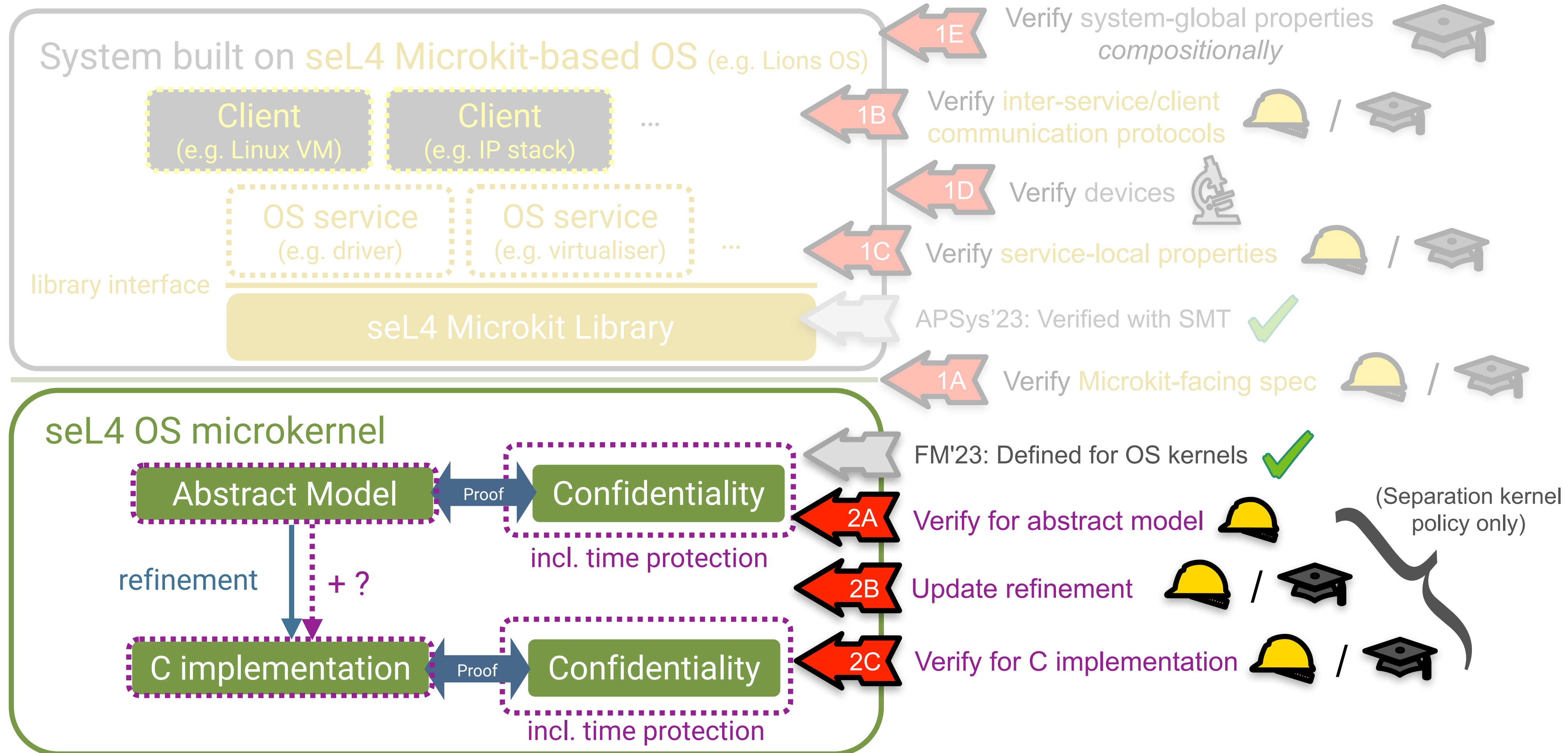
# Verification status

Microkit-based OS Services



# Verification status

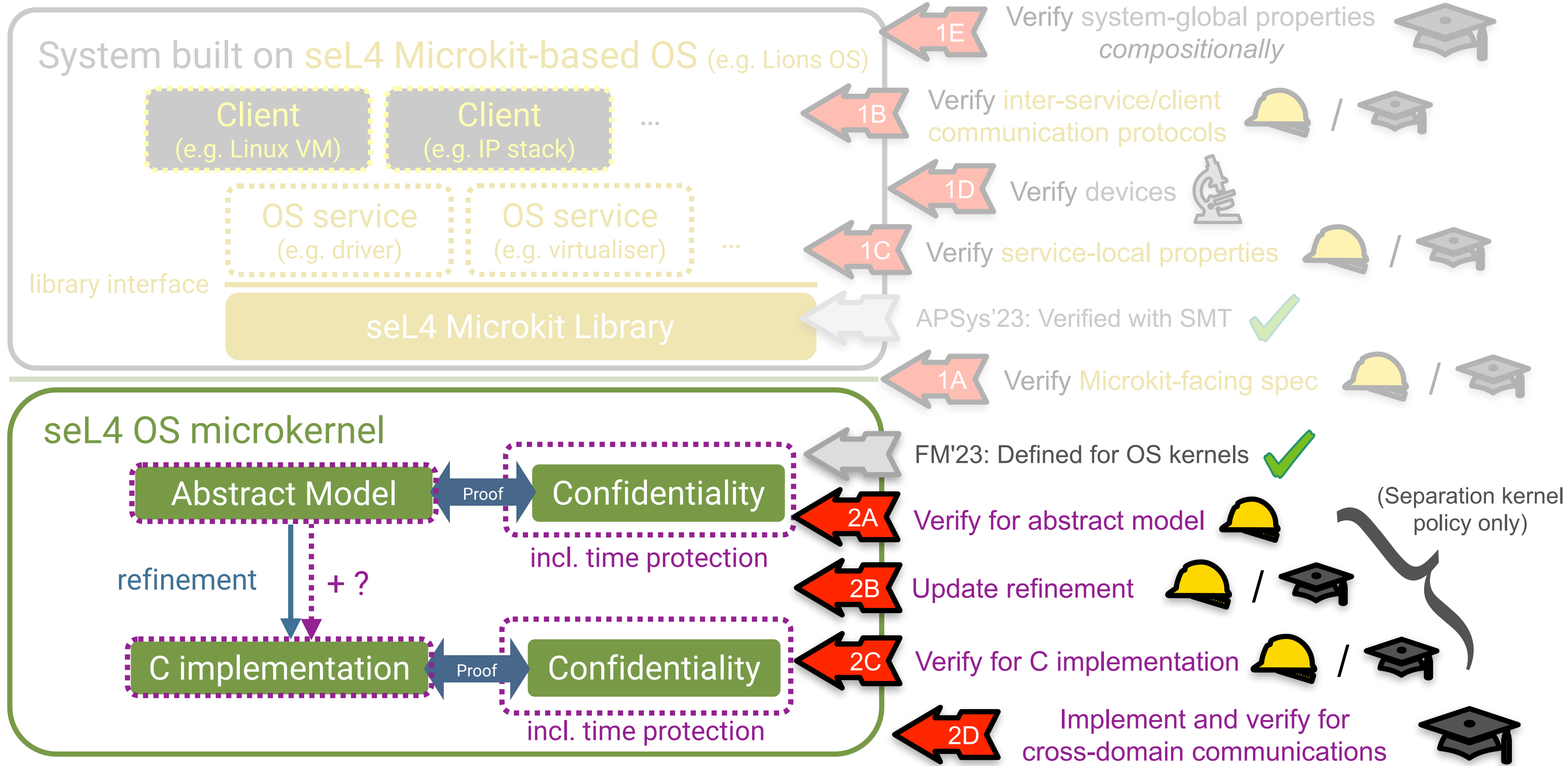
## Microkit-based OS Services



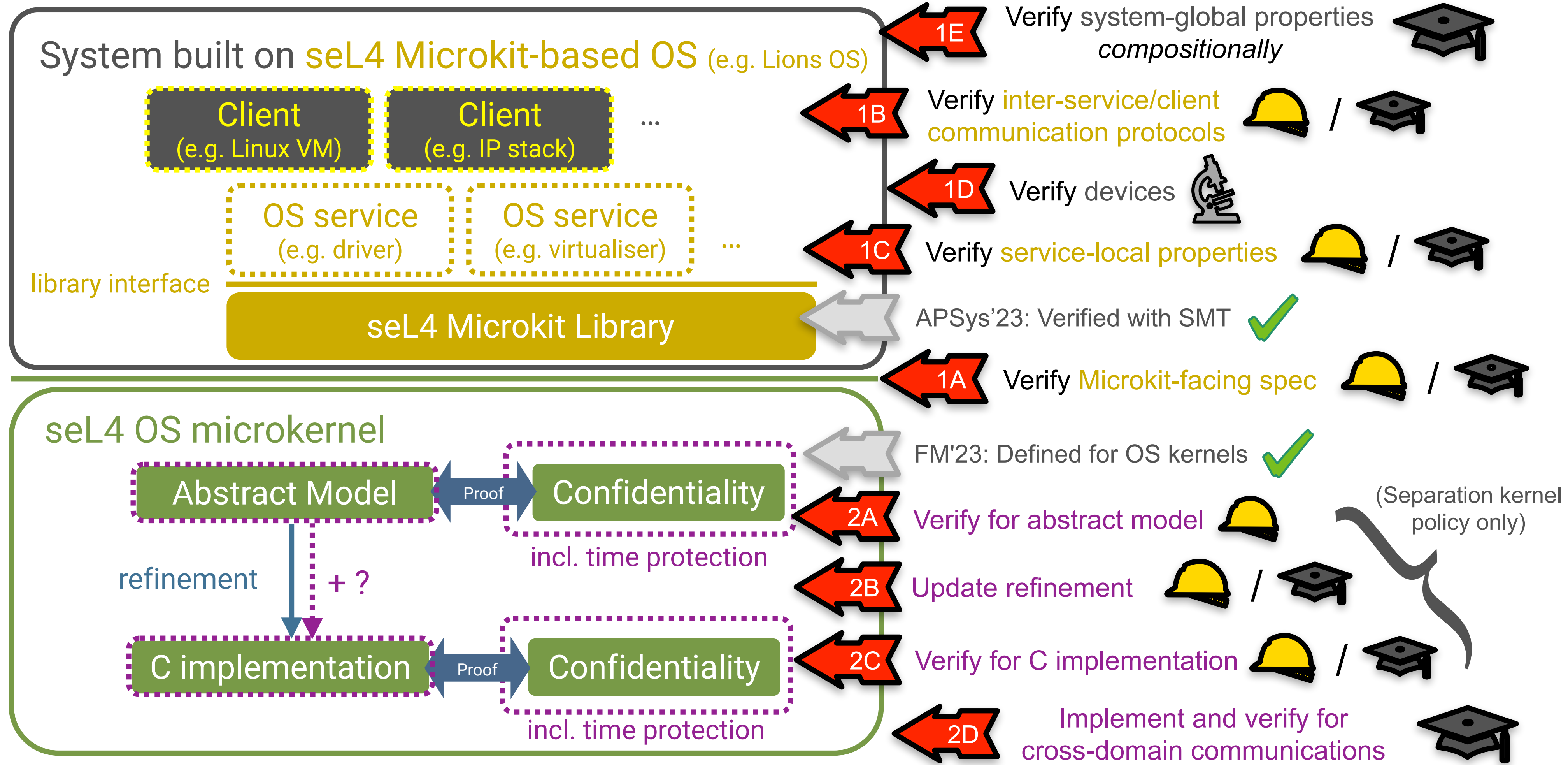
## Time Protection

# Verification status

Microkit-based OS Services



## Microkit-based OS Services



# Trustworthy Systems @ CSE, UNSW Sydney

