ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

CAPSTONE: A Capability-based Foundation for Trustless Secure Memory Access

32nd USENIX Security Symposium

Jason Zhijingcheng Yu, Conrad Watt, Aditya Badole,
Trevor E. Carlson, Prateek Saxena

National University of Singapore
University of Cambridge

NUS | Computing
National University of Singapore

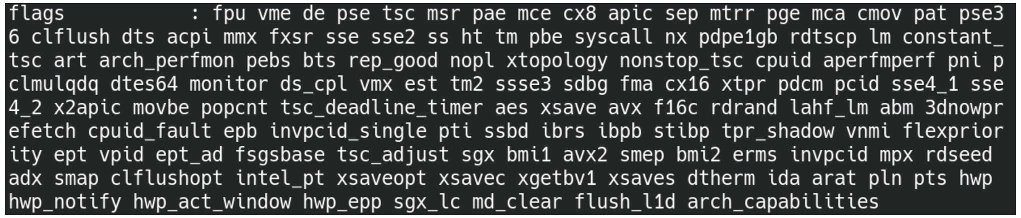
UNIVERSITY OF CAMBRIDGE

World of Security Extensions

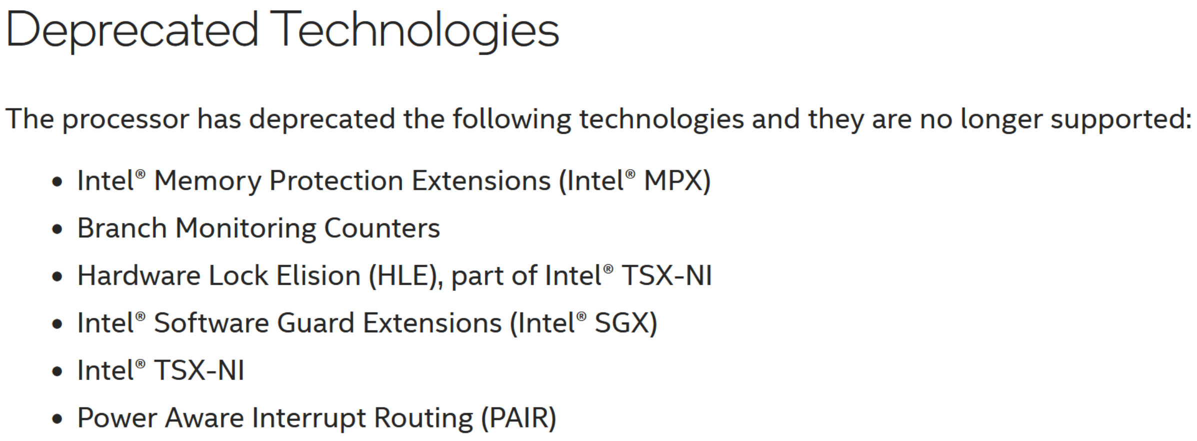
Pointer Integrity	[ARMv8 Pointer Authentication Code]
Spatial Memory Safety	[Intel MPK, x86/64 DEP/NX] [Intel MPX, RISC-V/ARM CHERI]
Temporal Memory Safety	[None]
Concurrent Thread Safety	[Intel TSX – Transactional Synchronization Extensions]
Intra-process Sandboxing	[Intel SGX] [x86 Segmentation]
Process Sandboxing	[x86/64 Privilege Rings]
Virtualization	[AMD SEV] [Intel VT-x] [Intel TDX] [ARM CCA]
Red-Green Secure Worlds	[ARM TZ] [Intel TXT]
Nested / App Virtualization	[Intel VT-x] [Intel SGX]

Problems with Security Extensions

1. Unreliable availability of security features



```
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse3
6 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_
tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni p
clmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse
4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowpr
efetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexprior
ity ept vpid ept_ad fsgsbase tsc_adjust sgx bmi1 avx2 smep bmi2 erms invpcid mpx rdseed
adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp
hwp_notify hwp_act_window hwp_epp sgx_lc md_clear flush_l1d arch_capabilities
```



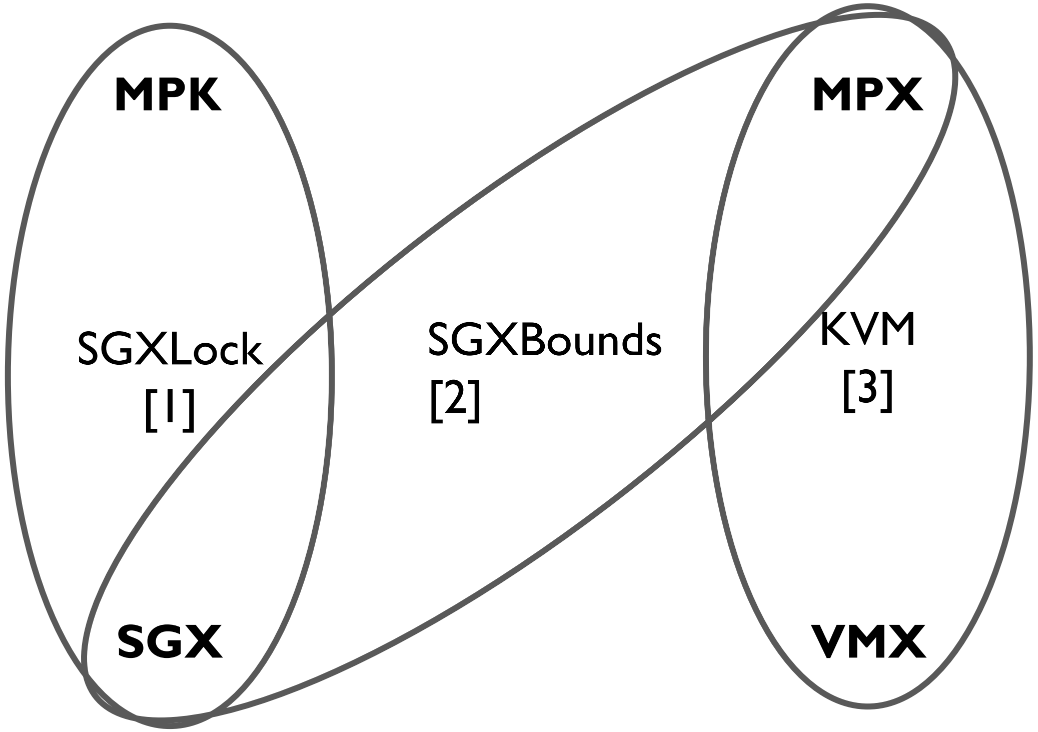
Deprecated Technologies

The processor has deprecated the following technologies and they are no longer supported:

- Intel® Memory Protection Extensions (Intel® MPX)
- Branch Monitoring Counters
- Hardware Lock Elision (HLE), part of Intel® TSX-NI
- Intel® Software Guard Extensions (Intel® SGX)
- Intel® TSX-NI
- Power Aware Interrupt Routing (PAIR)

Source: https://edc.intel.com/content/www/us/en/design/ipla/software-development-platforms/client/platforms/alder-lake-desktop/12th-generation-intel-core-processors-datasheet-volume-1-of-2/010/deprecated-technologies/ accessed 30 July 2023

2. Poor interoperability for multiple security goals



[1]
Y. Chen *et al.*, 'SGXLock: Towards Efficiently Establishing Mutual Distrust Between Host Application and Enclave for SGX', in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds., USENIX Association, 2022, pp. 4129–4146. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/chen-yuan

[2]
D. Kuvaiskii *et al.*, 'SGXBOUNDS: Memory Safety for Shielded Execution', in *Proceedings of the Twelfth European Conference on Computer Systems*, Belgrade Serbia: ACM, Apr. 2017, pp. 205–221. doi: 10.1145/3064176.3064192.

Problems with Security Extensions

1. Unreliable availability of security features

```
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse3
6 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_
tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni p
clmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse
4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowpr
efetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexprior
ity ept vpid ept_ad fsgsbase tsc_adjust sgx bmi1 avx2 smep bmi2 erms invpcid mpx rdseed
adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp
hwp_notify hwp_act_window hwp_epp sgx_lc md_clear flush_l1d arch_capabilities
```

Deprecated Technologies

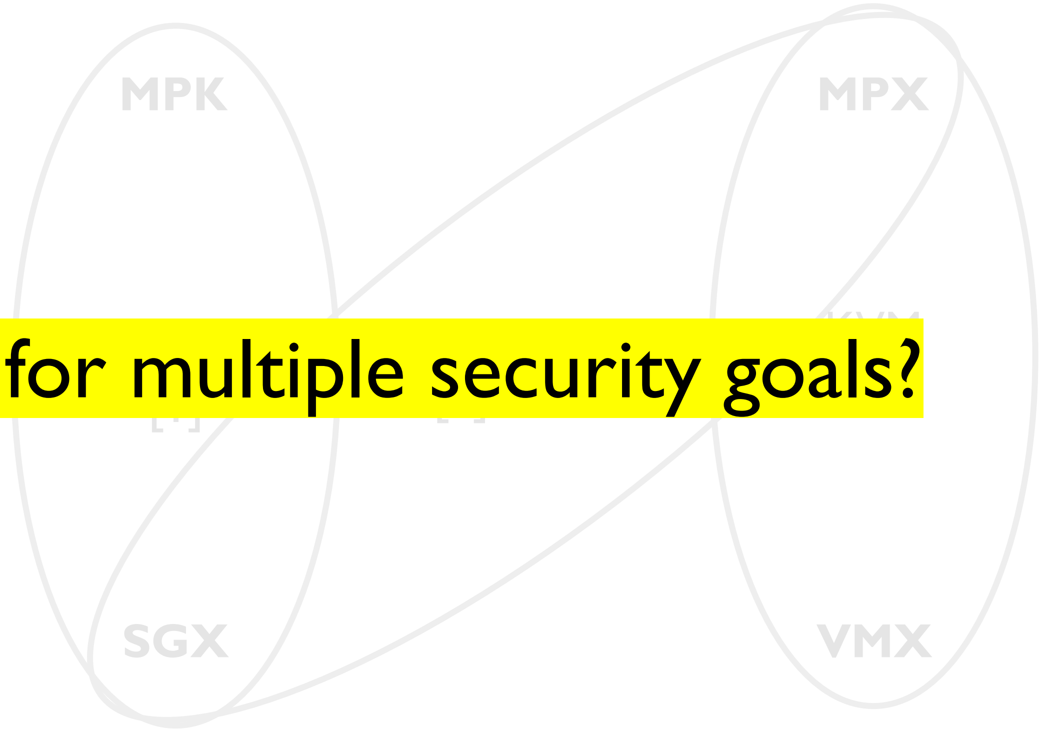
The proces

Is there a unified foundation for multiple security goals?

- Intel® Memory Protection Extensions (Intel® MPX)
- Branch Monitoring Counters
- Hardware Lock Elision (HLE), part of Intel® TSX-NI
- Intel® Software Guard Extensions (Intel® SGX)
- Intel® TSX-NI
- Power Aware Interrupt Routing (PAIR)

Source: https://edc.intel.com/content/www/us/en/design/ipla/software-development-platforms/client/platforms/alder-lake-desktop/12th-generation-intel-core-processors-datasheet-volume-1-of-2/010/deprecated-technologies/ accessed 30 July 2023

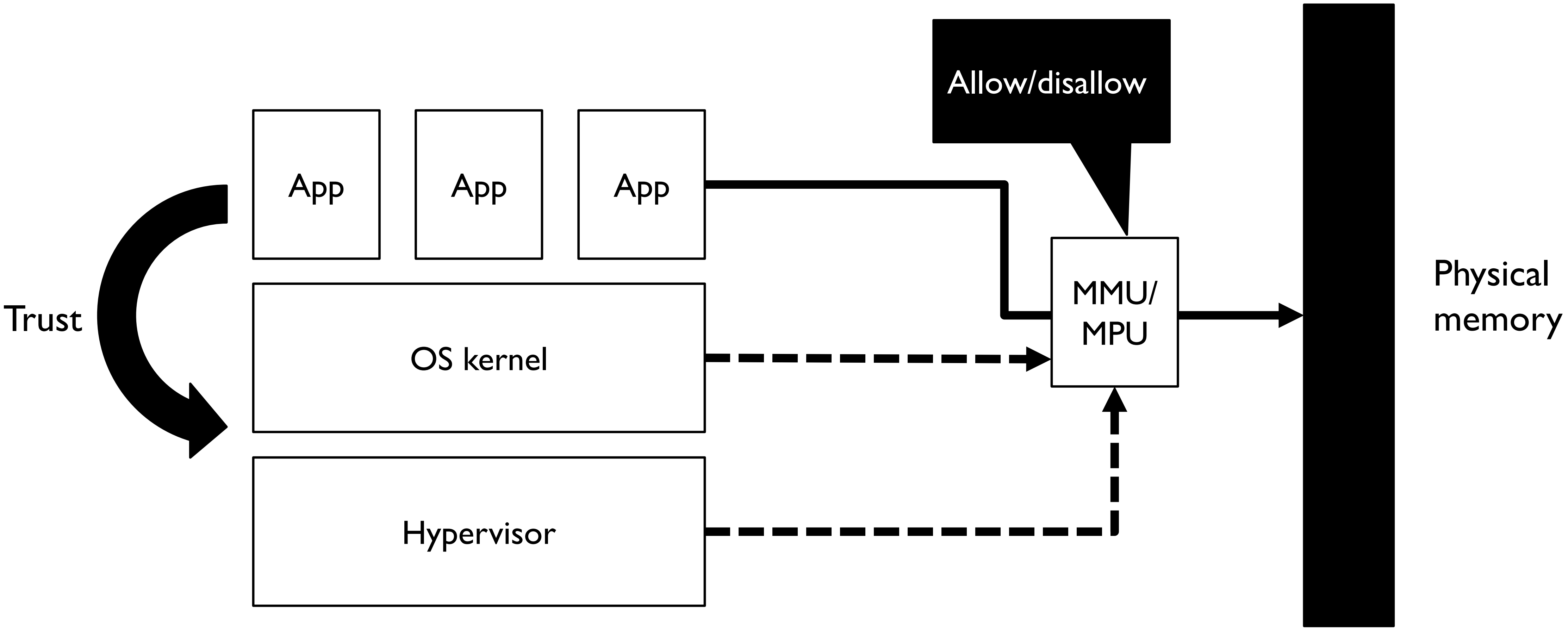
2. Poor interoperability for multiple security goals



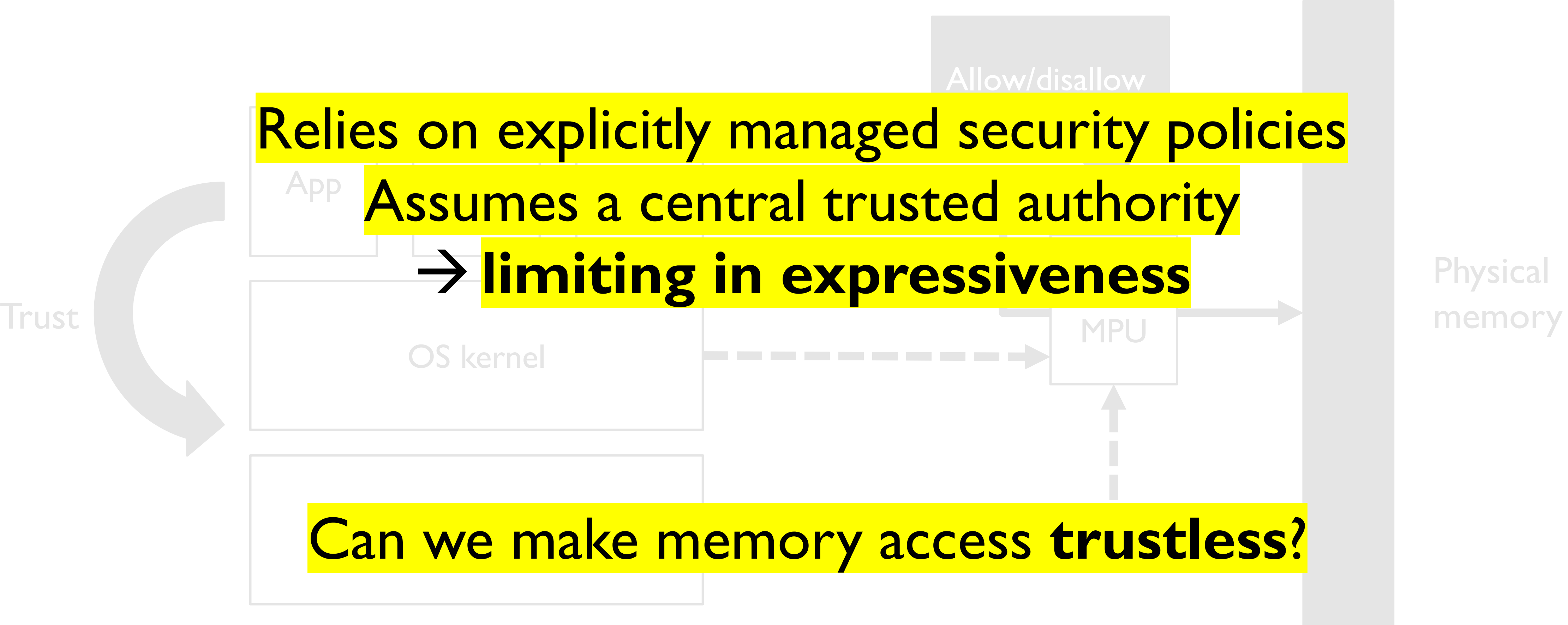
[1]
Y. Chen *et al.*, 'SGXLock: Towards Efficiently Establishing Mutual Distrust Between Host Application and Enclave for SGX', in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds., USENIX Association, 2022, pp. 4129–4146. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/chen-yuan

[2]
D. Kuvaiskii *et al.*, 'SGXBOUNDS: Memory Safety for Shielded Execution', in *Proceedings of the Twelfth European Conference on Computer Systems*, Belgrade Serbia: ACM, Apr. 2017, pp. 205–221. doi: 10.1145/3064176.3064192.

Traditional Architectures Rely on Access Control



Traditional Architectures Rely on Access Control



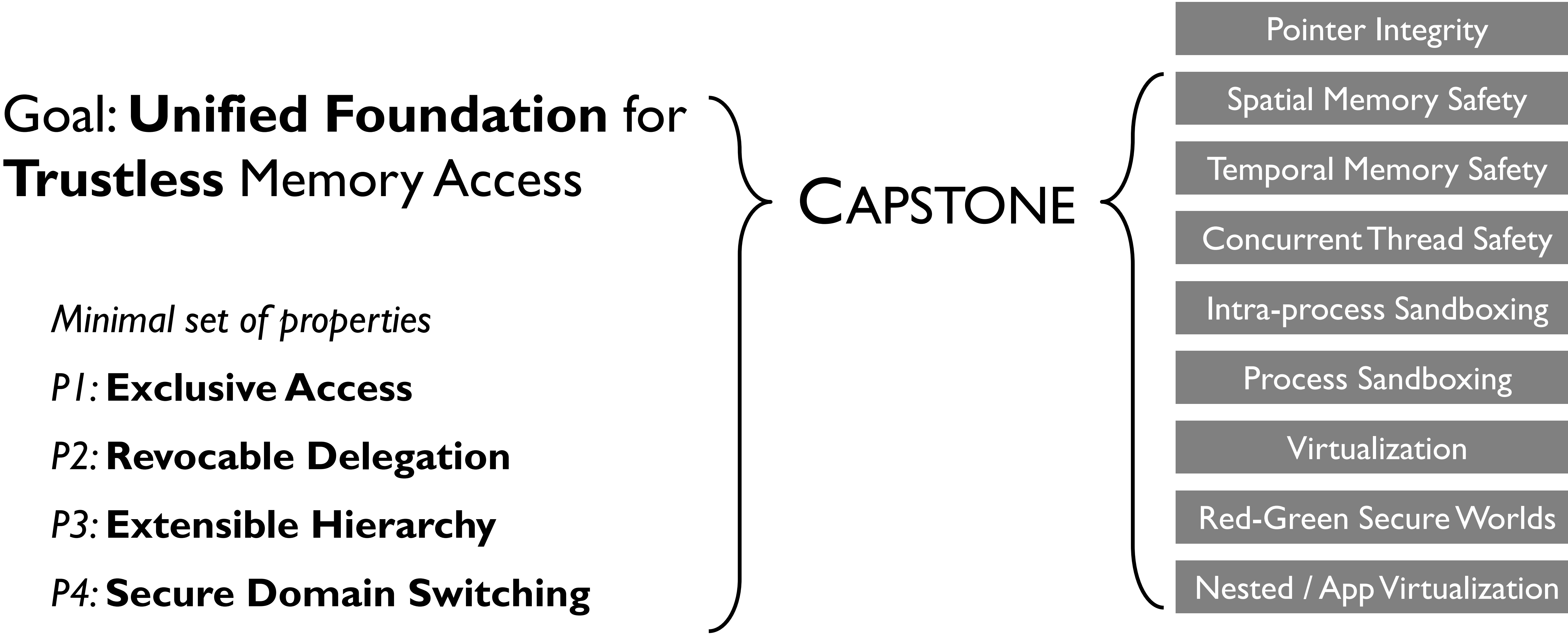
Relies on explicitly managed security policies

Assumes a central trusted authority

→ **limiting in expressiveness**

Can we make memory access **trustless**?

Contributions



Goal: **Unified Foundation** for **Trustless** Memory Access

Minimal set of properties

P1: **Exclusive Access**

P2: **Revocable Delegation**

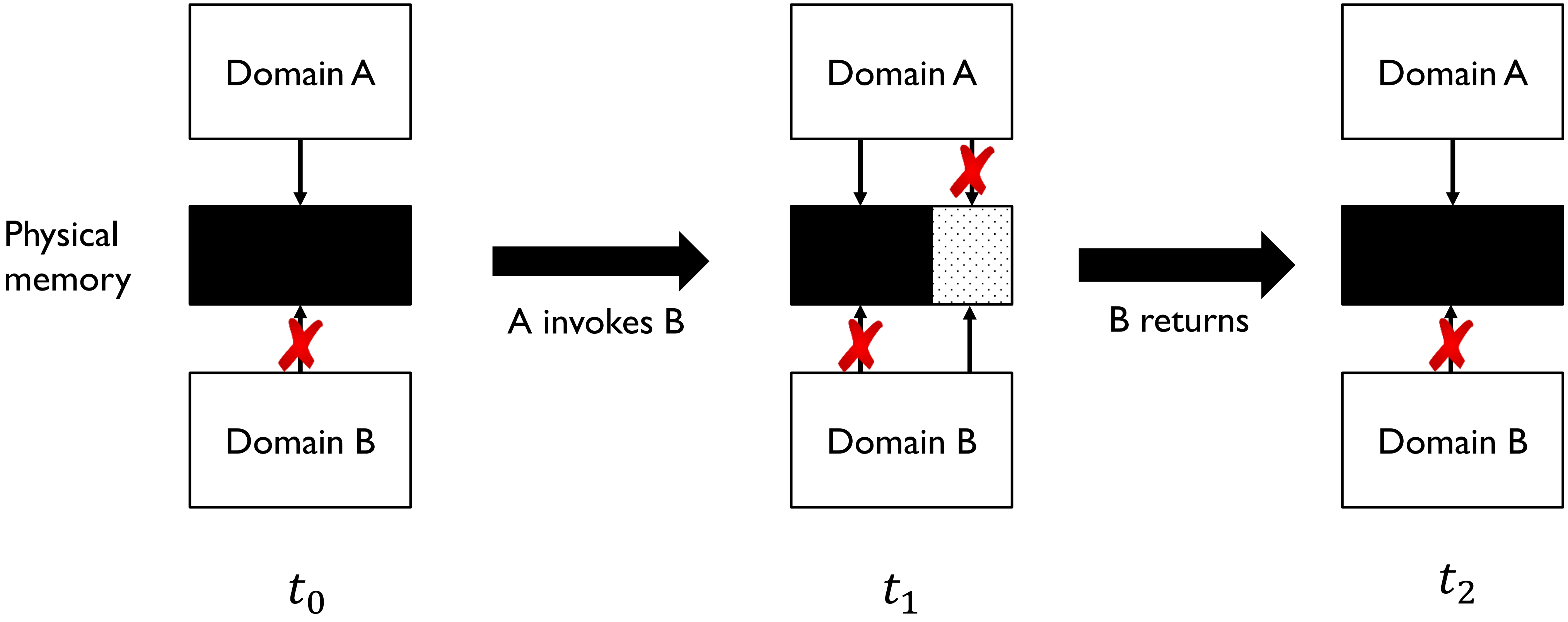
P3: **Extensible Hierarchy**

P4: **Secure Domain Switching**

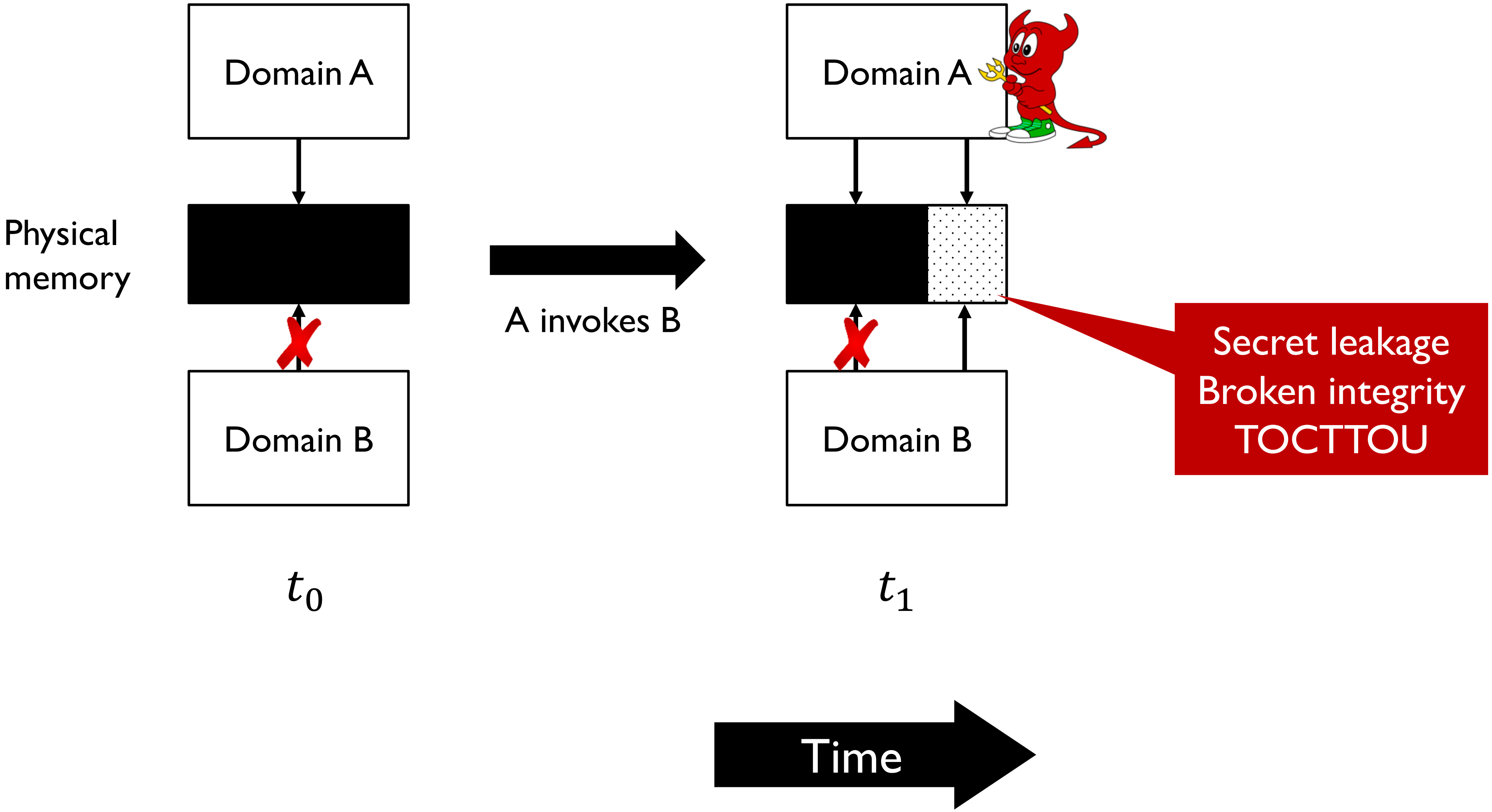
CAPSTONE

- Pointer Integrity
- Spatial Memory Safety
- Temporal Memory Safety
- Concurrent Thread Safety
- Intra-process Sandboxing
- Process Sandboxing
- Virtualization
- Red-Green Secure Worlds
- Nested / App Virtualization

Threat Model: Benign Scenario

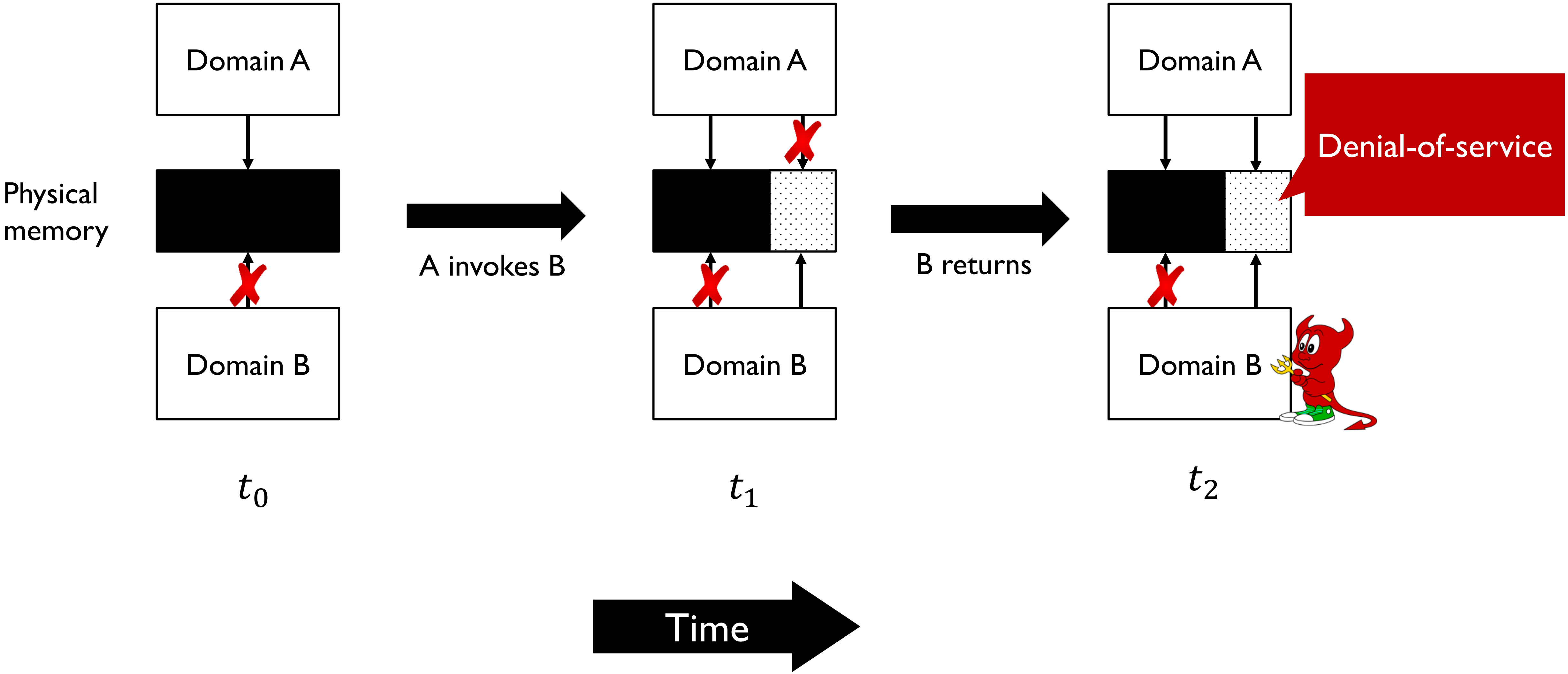


Threat Model: Malicious Scenario



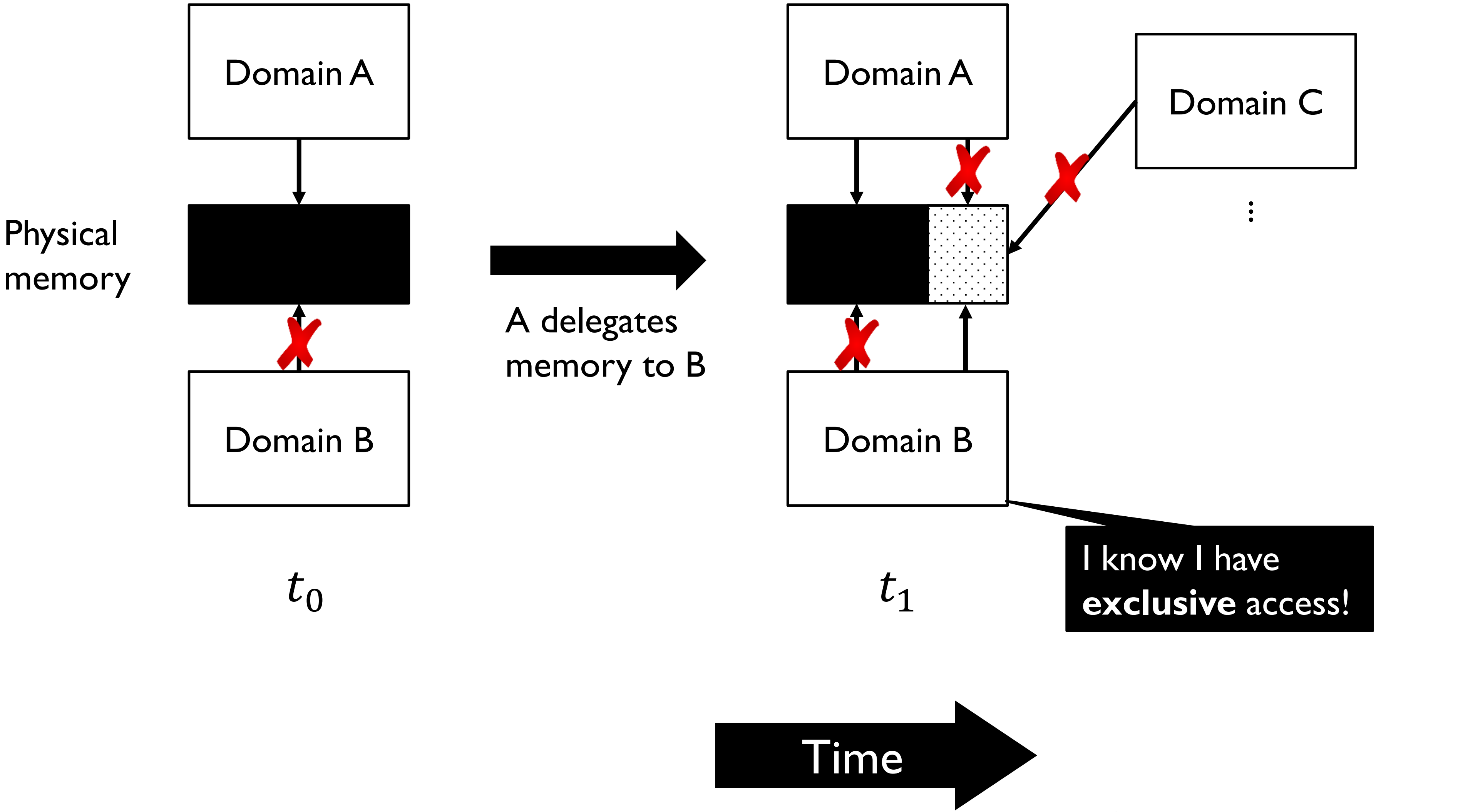
Secret leakage
Broken integrity
TOCTTOU

Threat Model: Malicious Scenario



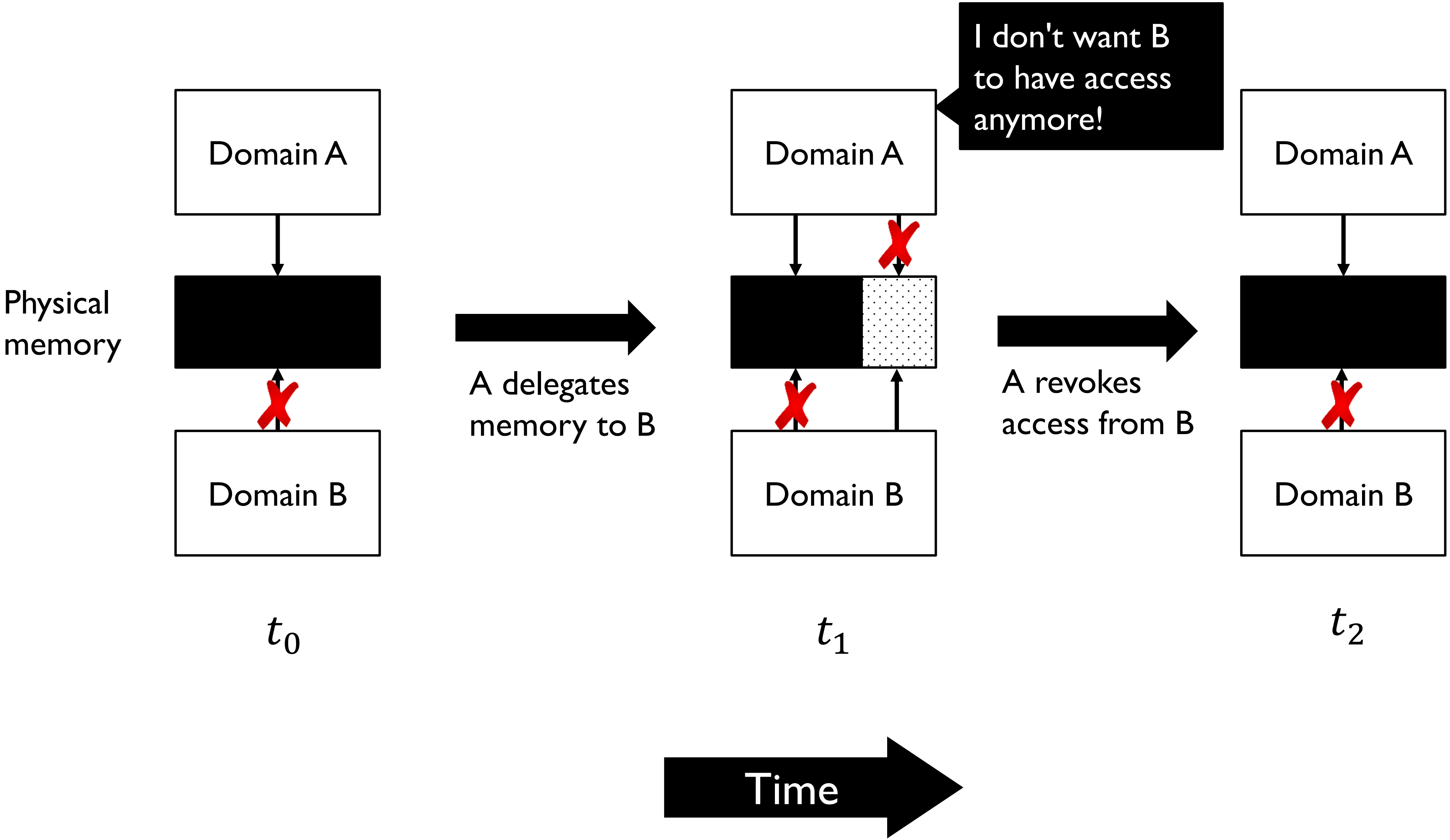
Minimal set of properties for a unified foundation

Property 1: Exclusive Access



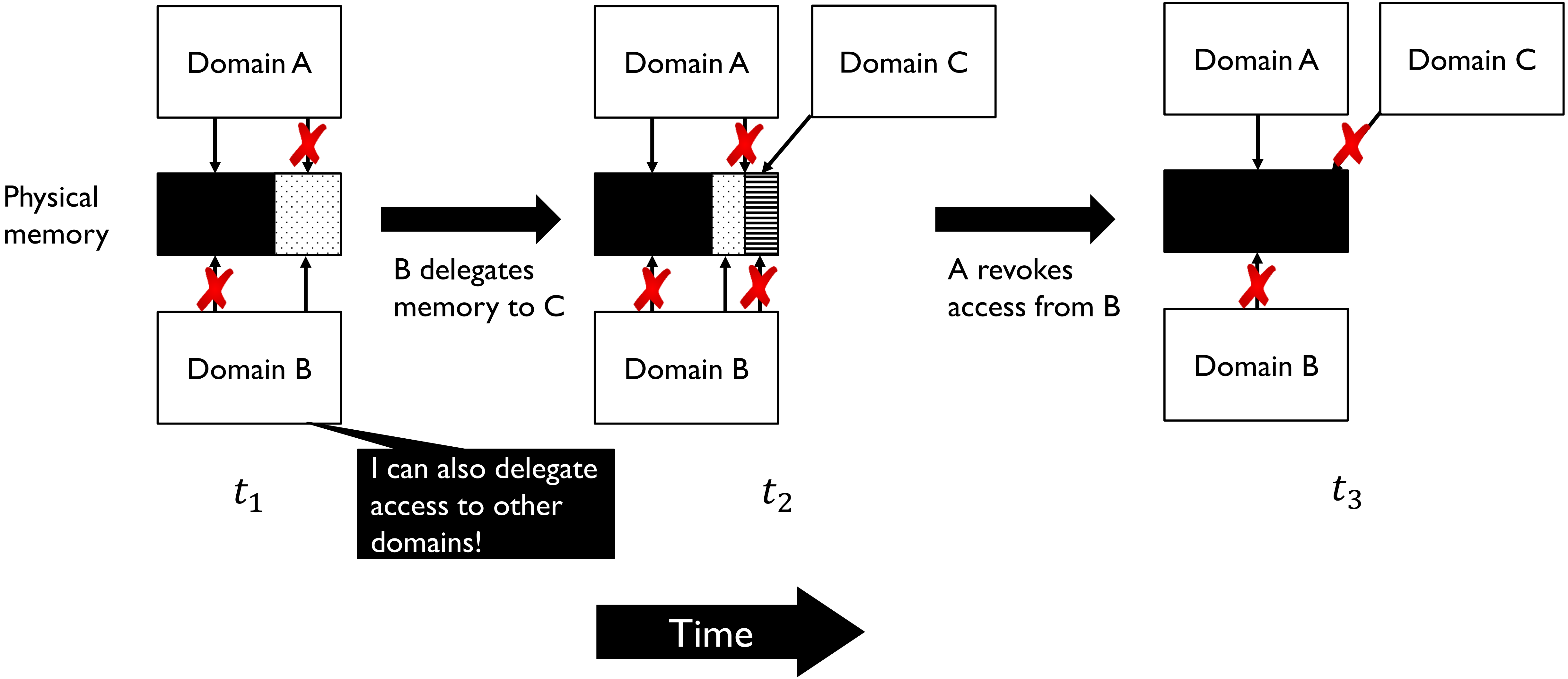
J. Z. Yu, S. Shinde, T. E. Carlson, and P. Saxena, 'Elasticlave: An Efficient Memory Model for Enclaves', in 31st USENIX Security Symposium

Property 2: Revocable Delegation



J. Z. Yu, S. Shinde, T. E. Carlson, and P. Saxena, 'Elasticlave: An Efficient Memory Model for Enclaves', in 31st USENIX Security Symposium

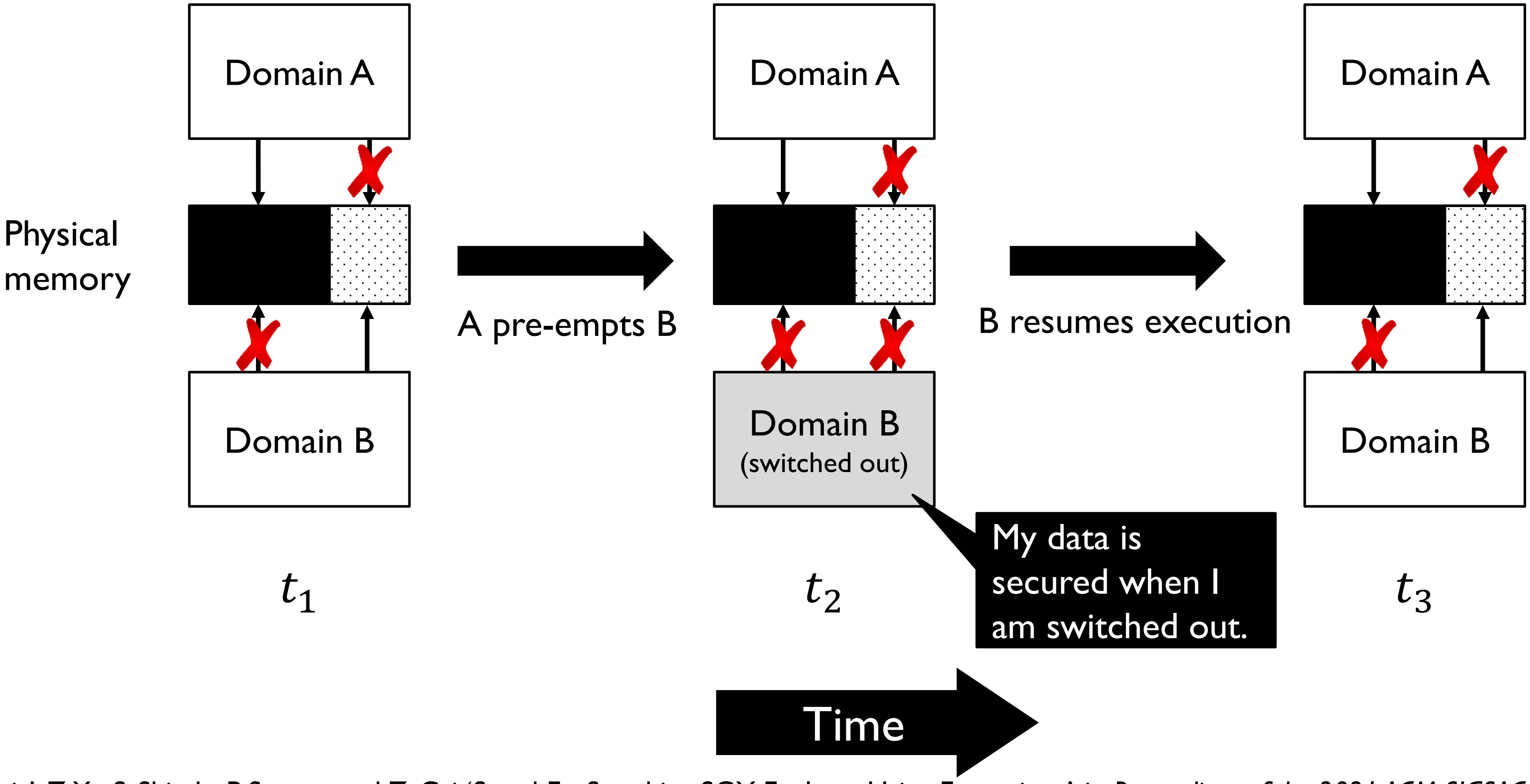
Property 3: Extensible Hierarchy



I can also delegate access to other domains!

J. Z. Yu, S. Shinde, T. E. Carlson, and P. Saxena, ‘Elasticlave: An Efficient Memory Model for Enclaves’, in 31st USENIX Security Symposium

Property 4: Secure Domain Switching



My data is secured when I am switched out.

J. Cui, J. Z. Yu, S. Shinde, P. Saxena, and Z. Cai, 'SmashEx: Smashing SGX Enclaves Using Exceptions', in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*

Properties for a Trustless Unified Foundation

P1: **Exclusive Access**

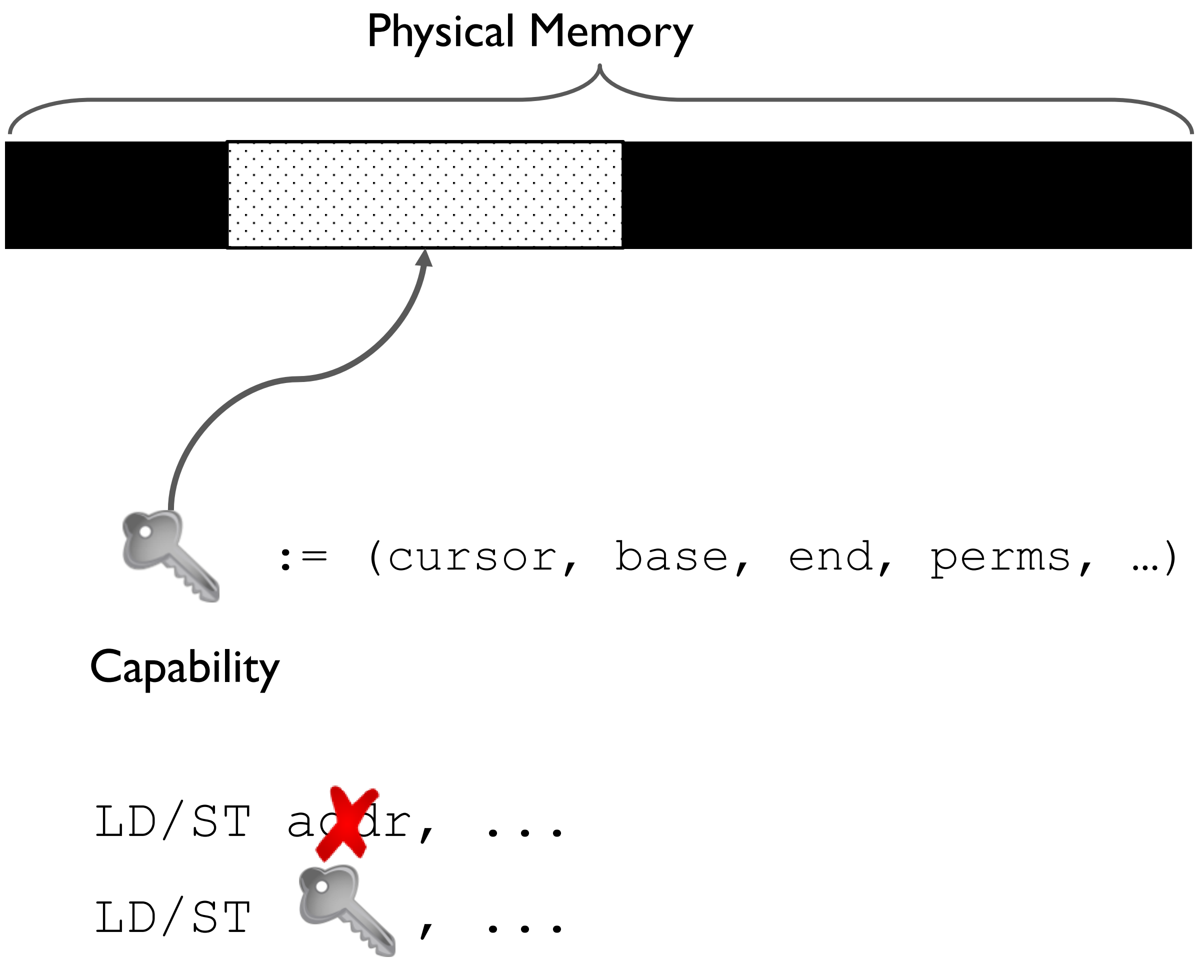
P2: **Revocable Delegation**

P3: **Extensible Hierarchy**

P4: **Secure Domain Switching**

How to enforce those properties through a unified interface?

Architectural Capabilities: A Baseline

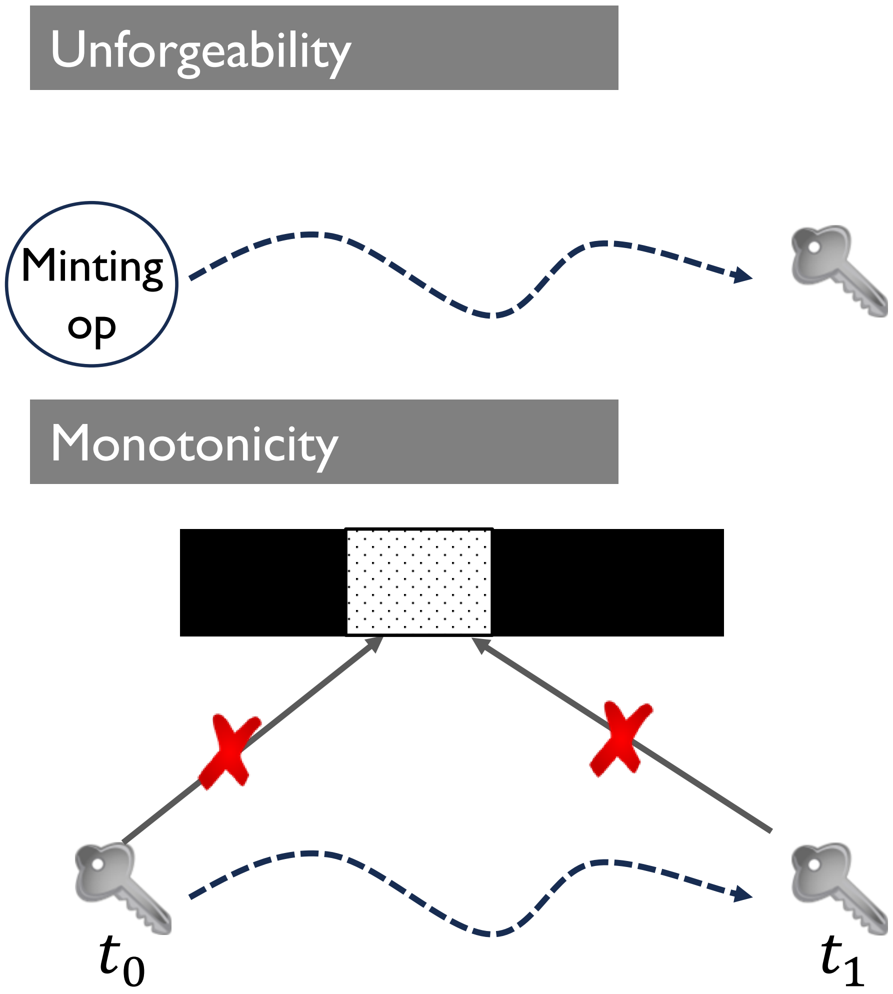


Physical Memory

:= (cursor, base, end, perms, …)

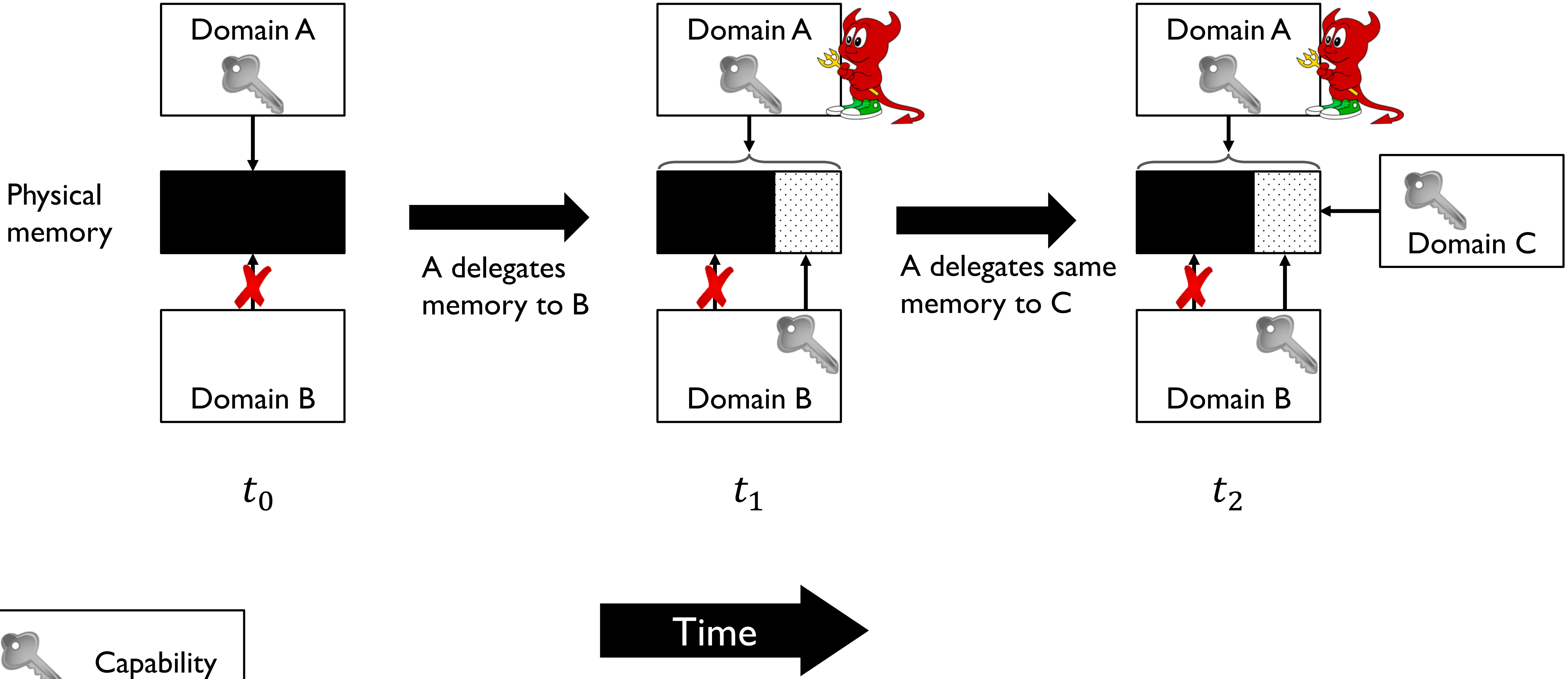
Capability

```
LD/ST addr, ...
LD/ST      , ...
```

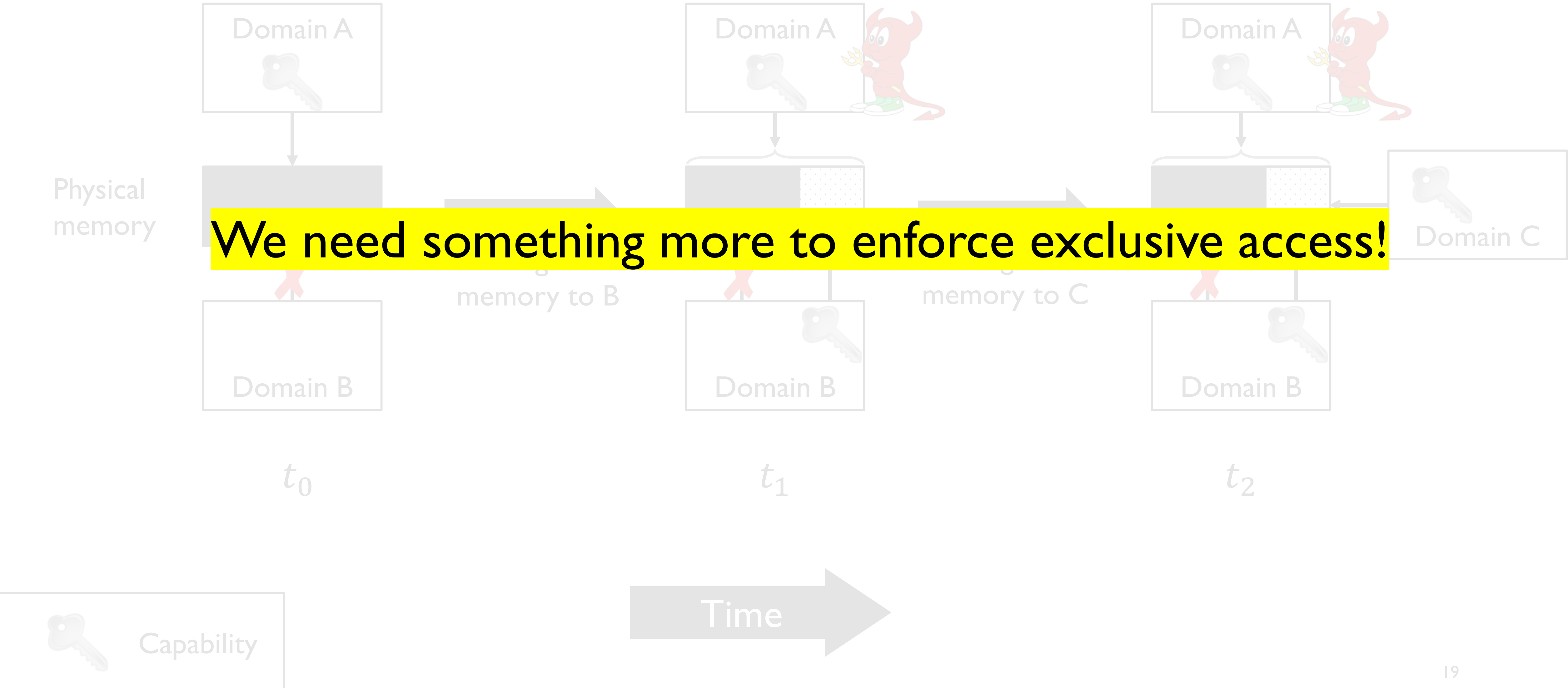


R. N. M. Watson *et al.*, ‘Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 8)’.

Enforcing Property 1: Exclusive Access

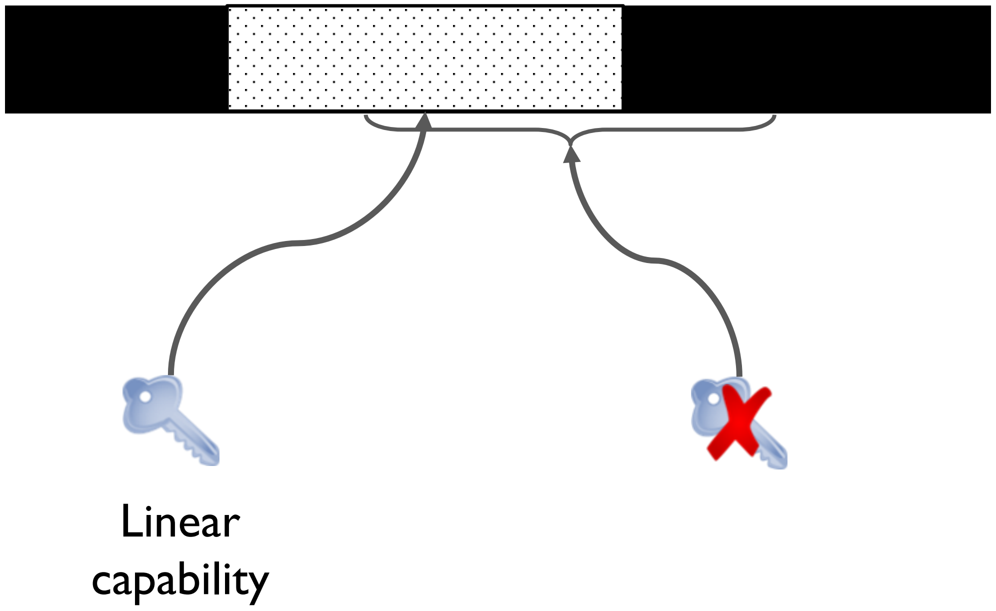


Enforcing Property 1: Exclusive Access



We need something more to enforce exclusive access!

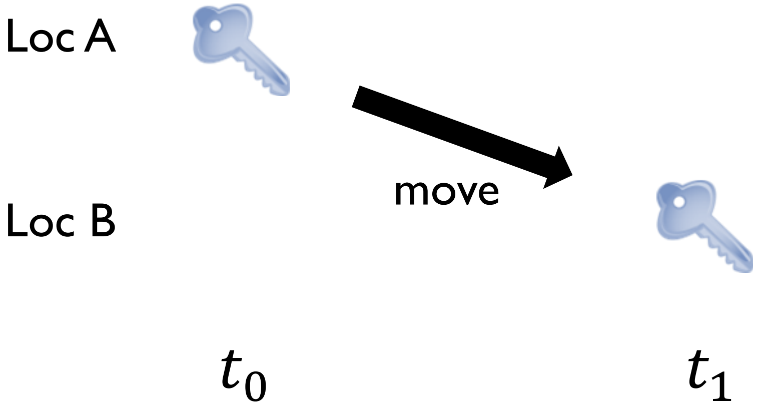
Exclusive Access: Linear Capabilities



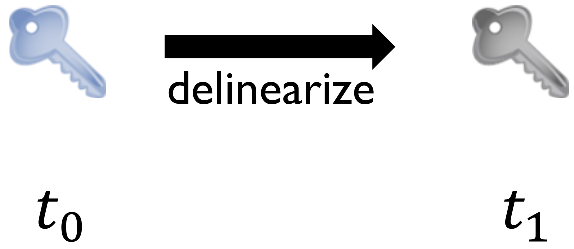
✓ Exclusive access

Linear Capability Operations

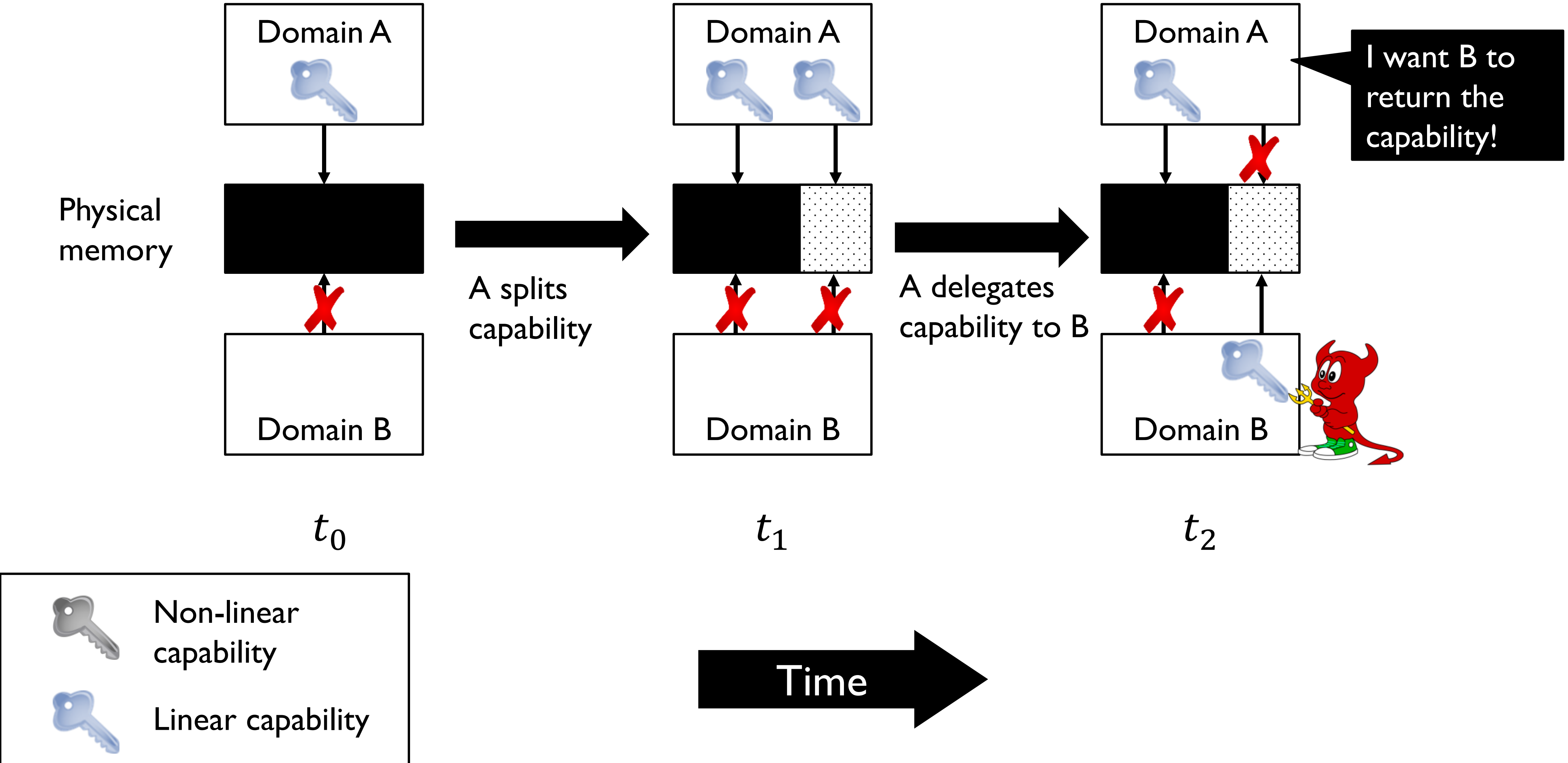
Move



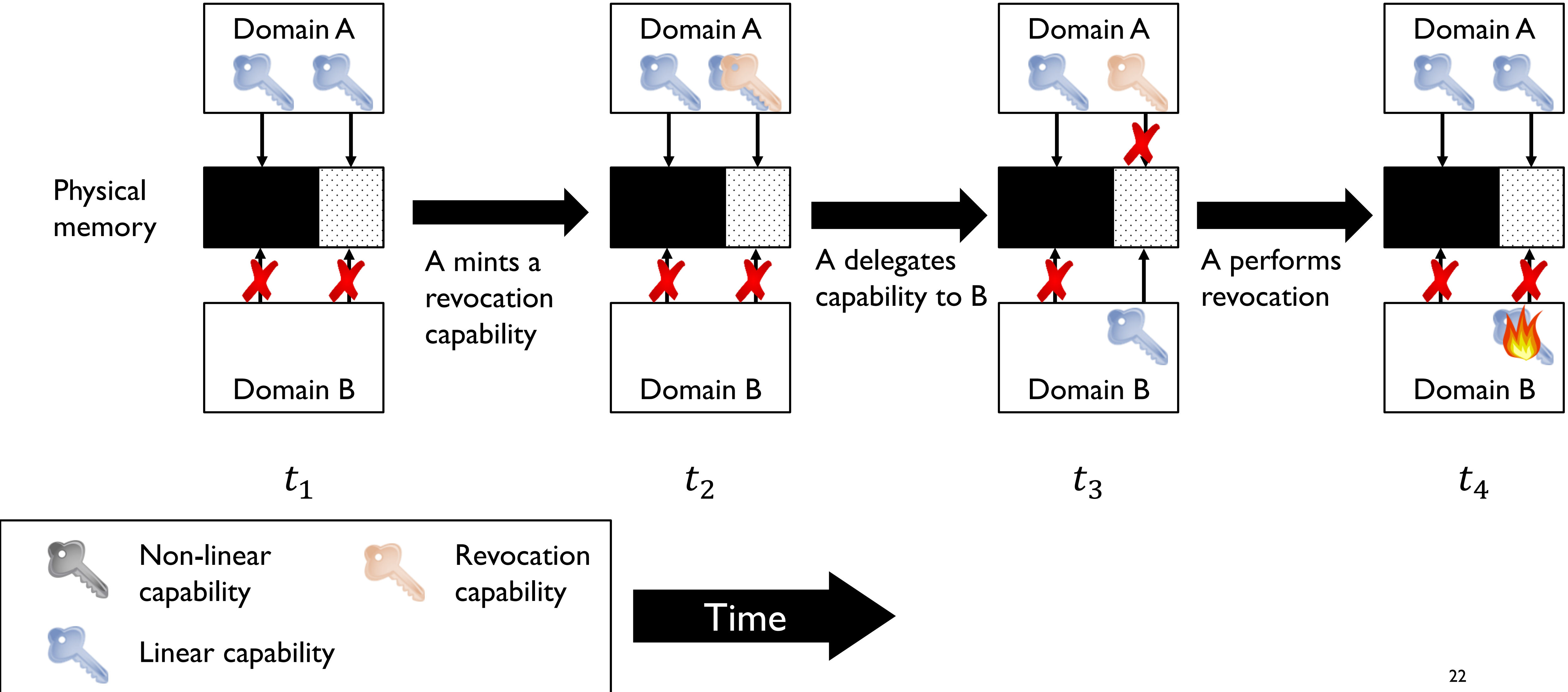
Delinearize



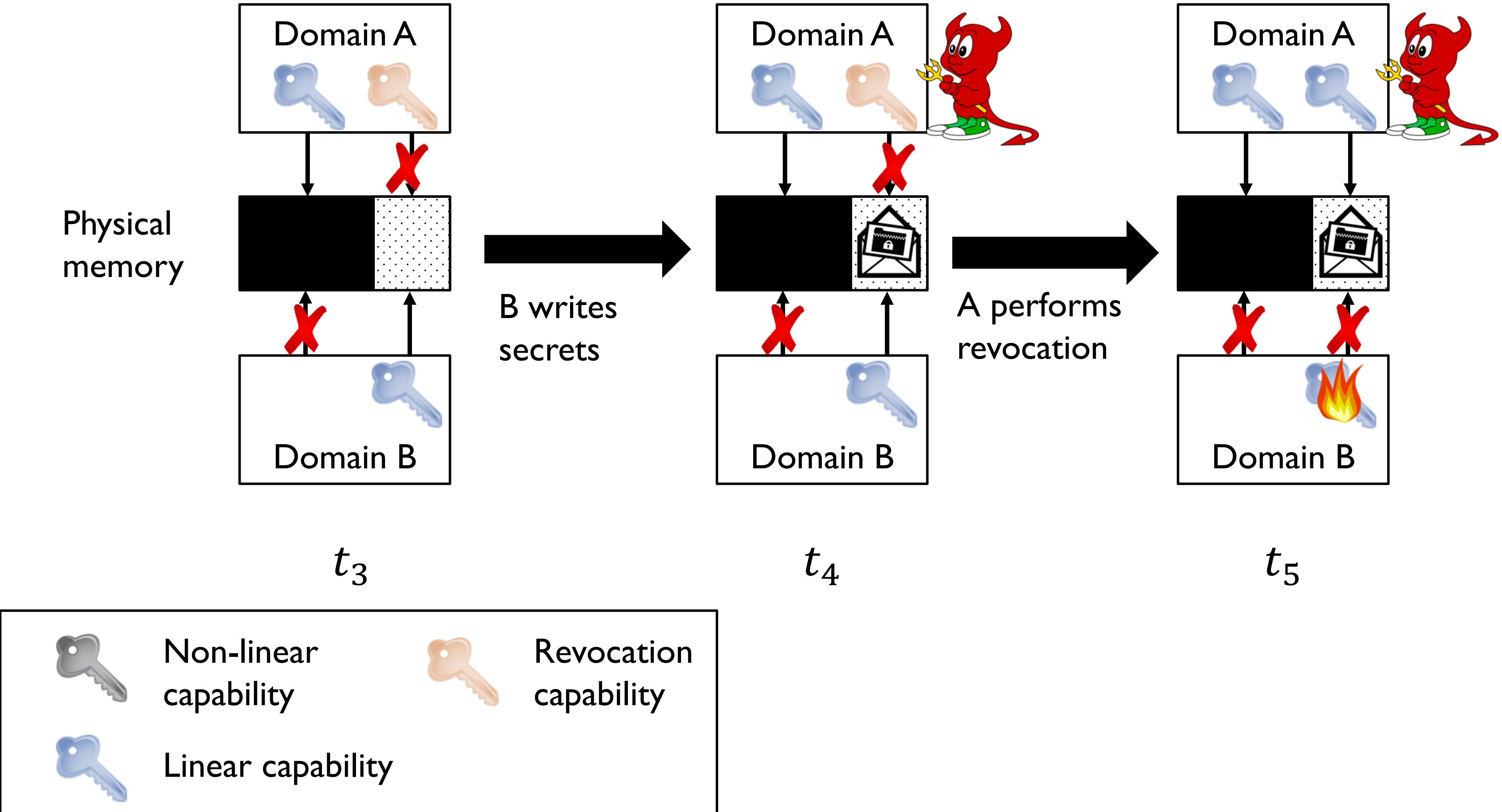
Memory Delegation with Linear Capabilities



Enforcing Property 2: Revocable Delegation



Problem: Secret Leakage Can Still Happen



Physical memory

Domain A

Domain B

t_3

B writes secrets

Domain A

Domain B

t_4

A performs revocation

Domain A

Domain B

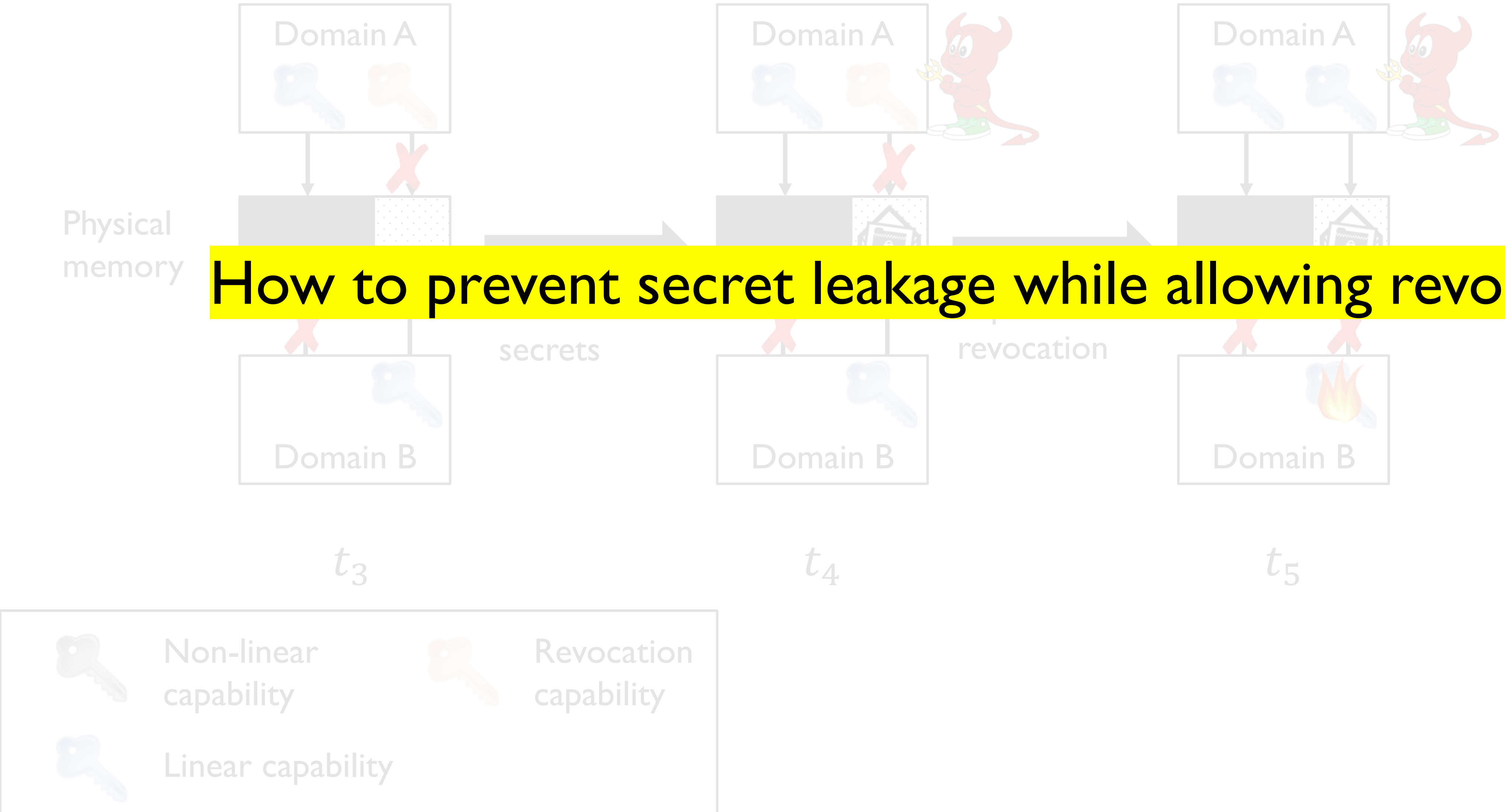
t_5

Non-linear capability

Revocation capability

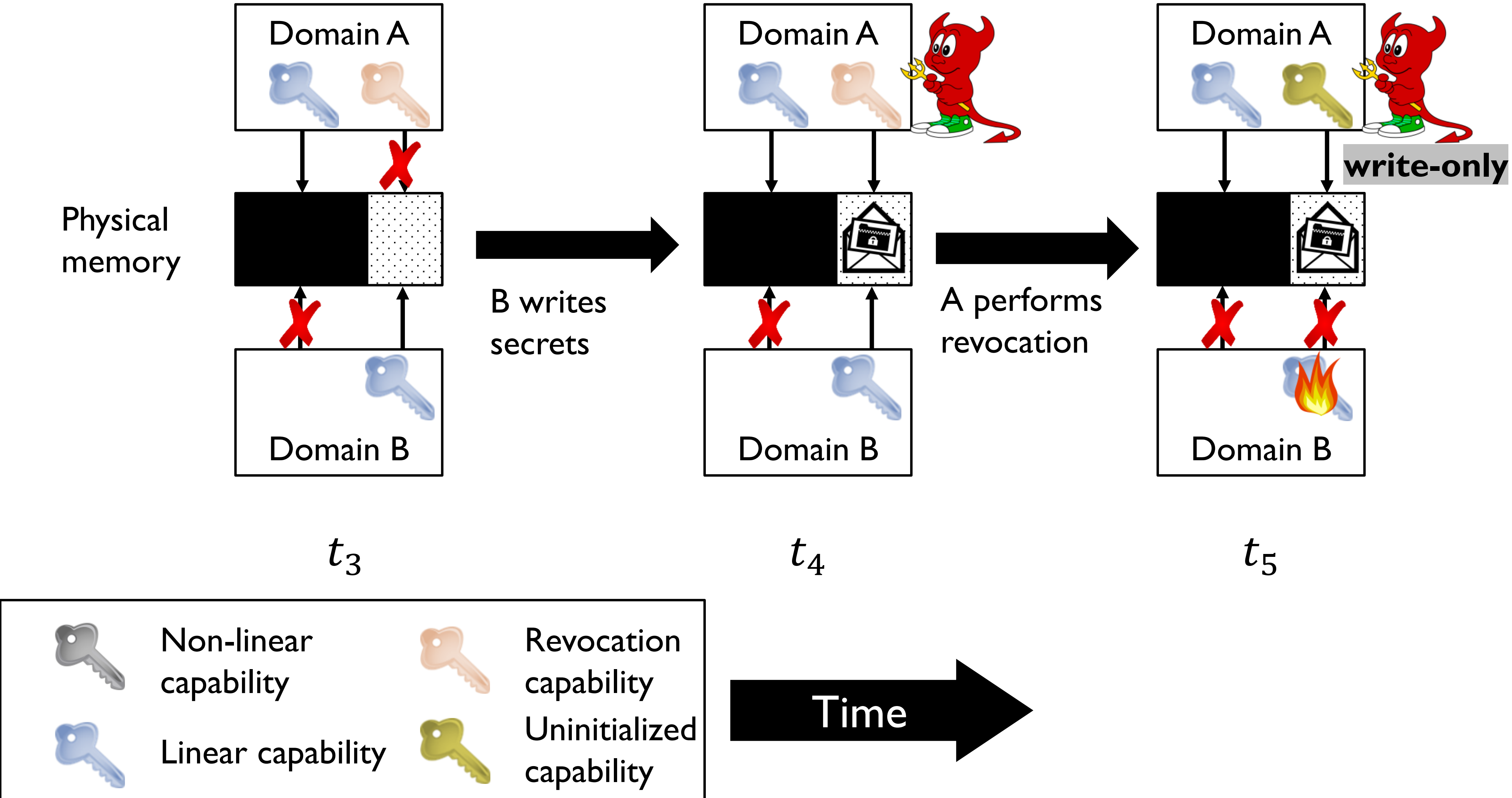
Linear capability

Problem: Secret Leakage Can Still Happen



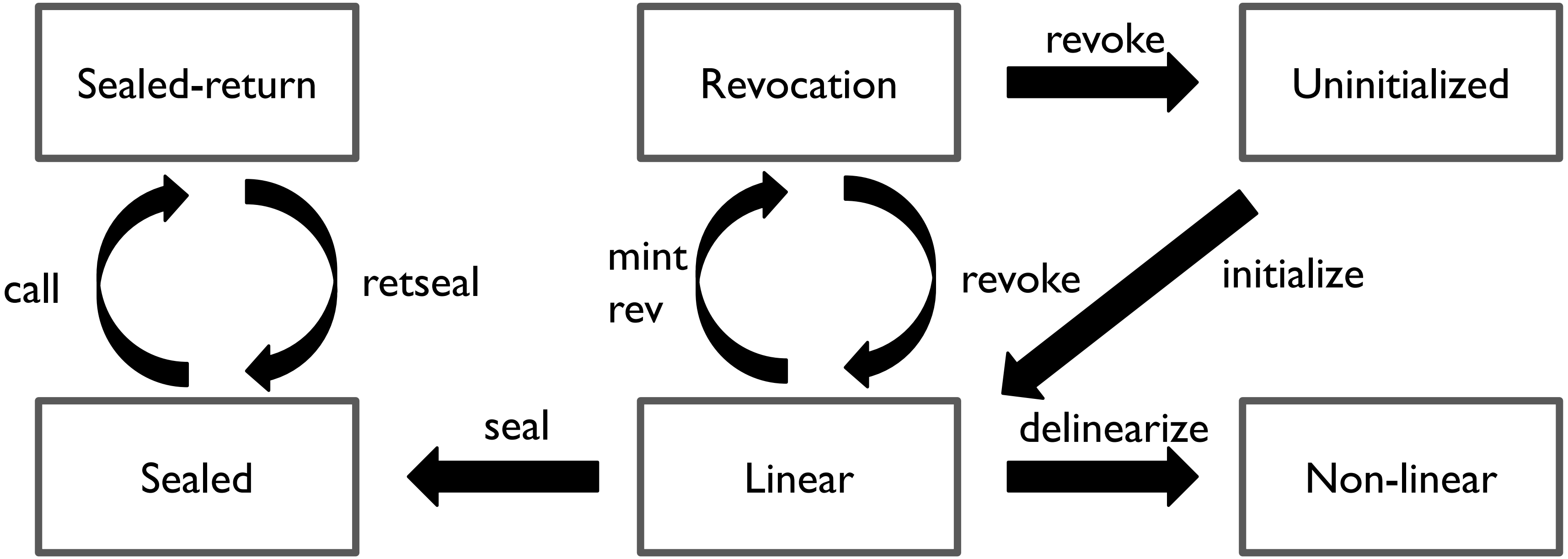
How to prevent secret leakage while allowing revocation?

Solution: Uninitialized Capabilities



CAPSTONE: Putting It Together

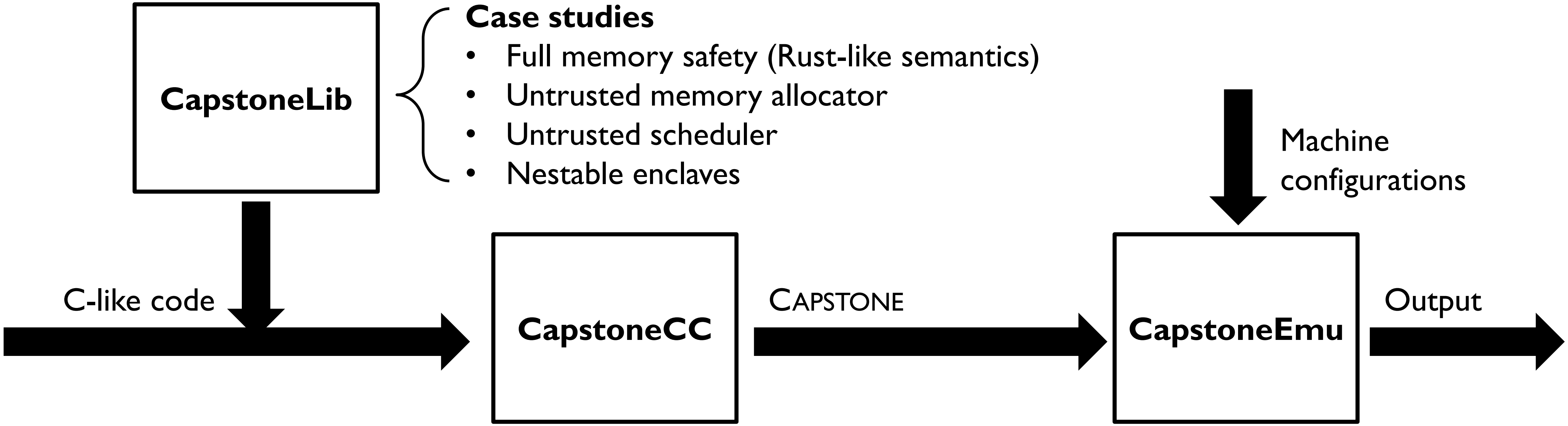
ISA with capability types and instructions



https://capstone.kisp-lab.org/

Implementation and Evaluation

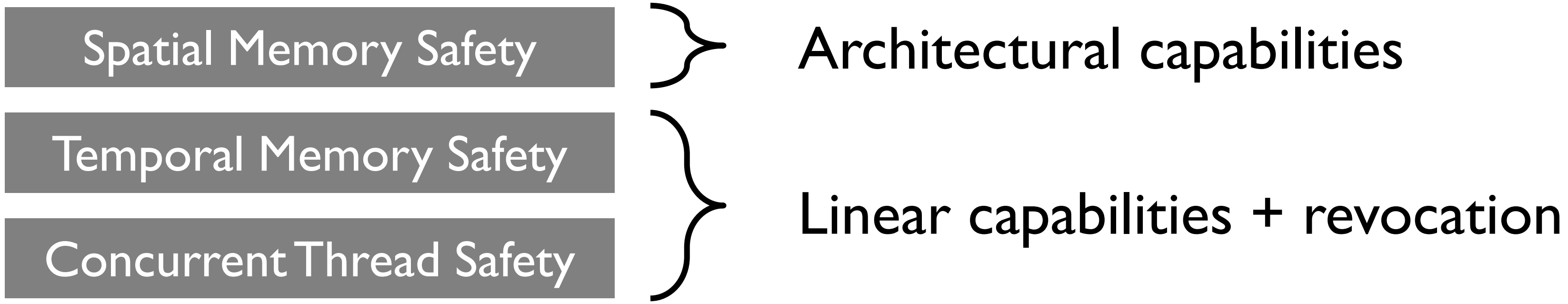
Functional Prototype



Case studies

- Full memory safety (Rust-like semantics)
- Untrusted memory allocator
- Untrusted scheduler
- Nestable enclaves

Full Memory Safety (Rust-like Semantics)



Spatial Memory Safety

Temporal Memory Safety

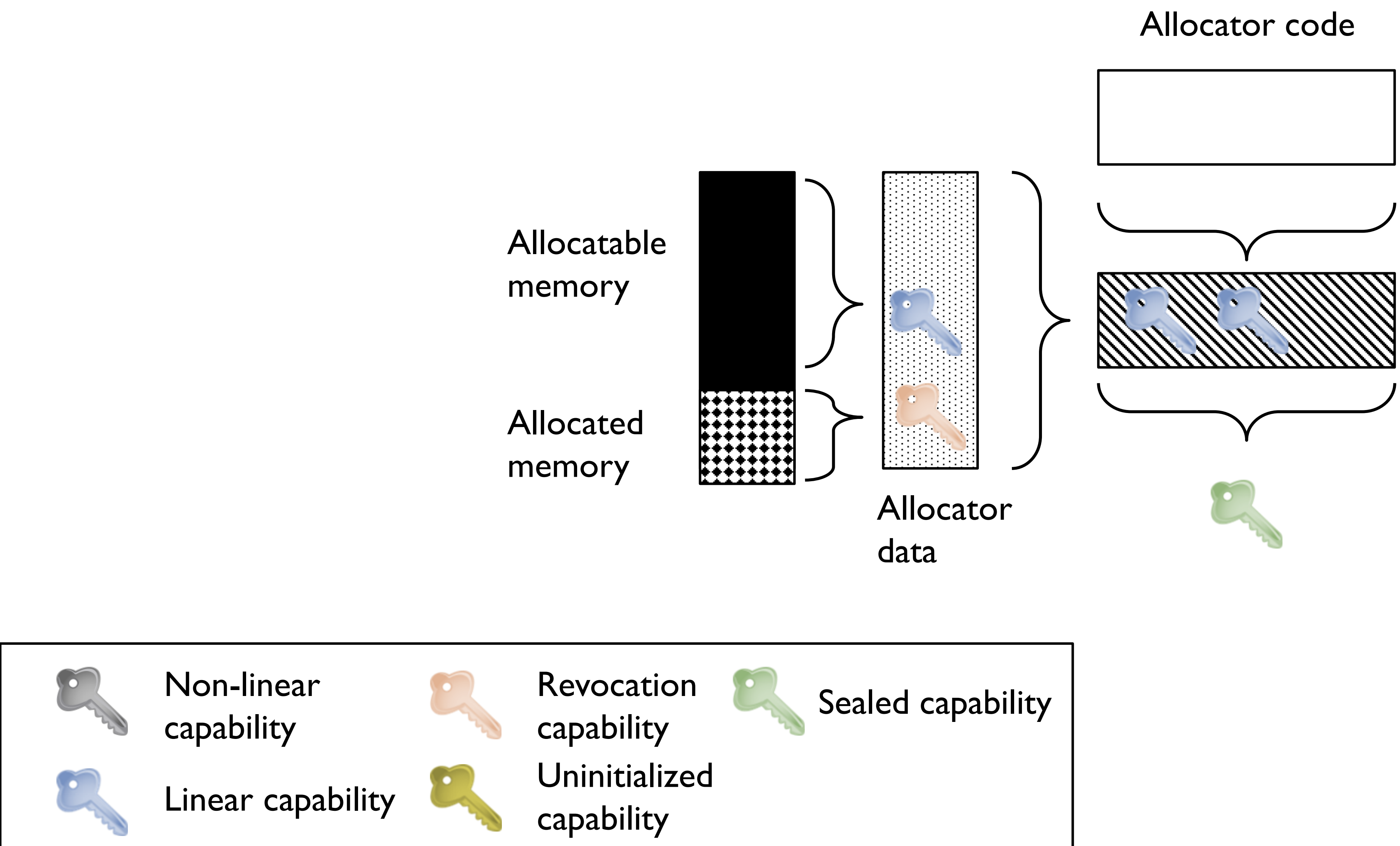
Concurrent Thread Safety

Architectural capabilities

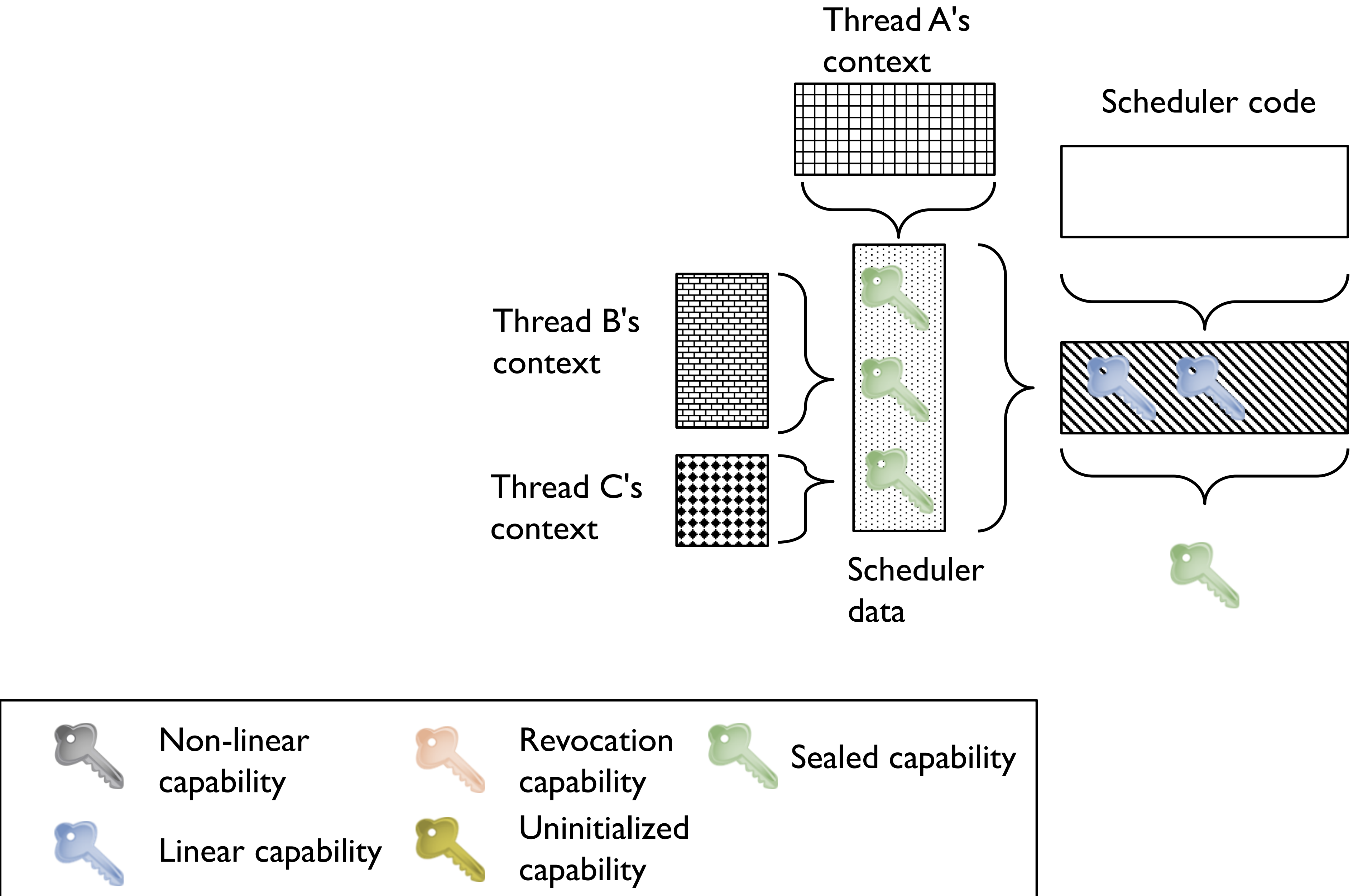
Linear capabilities + revocation

Operation	Rust semantics	Capstone
Move	`let a = b;`	`mov ra, rb;`
Immutable borrow	`let a = &b;`	`mrev rr, rb; delin rb; li r0, 0; tighten rb, r0; mov ra, rb; (use ra) revoke rr; mov rb, rr`
Mutable borrow	`let a = &mut b;`	`mrev rr, rb; mov ra, rb; (use ra) revoke rr; mov rb, rr`

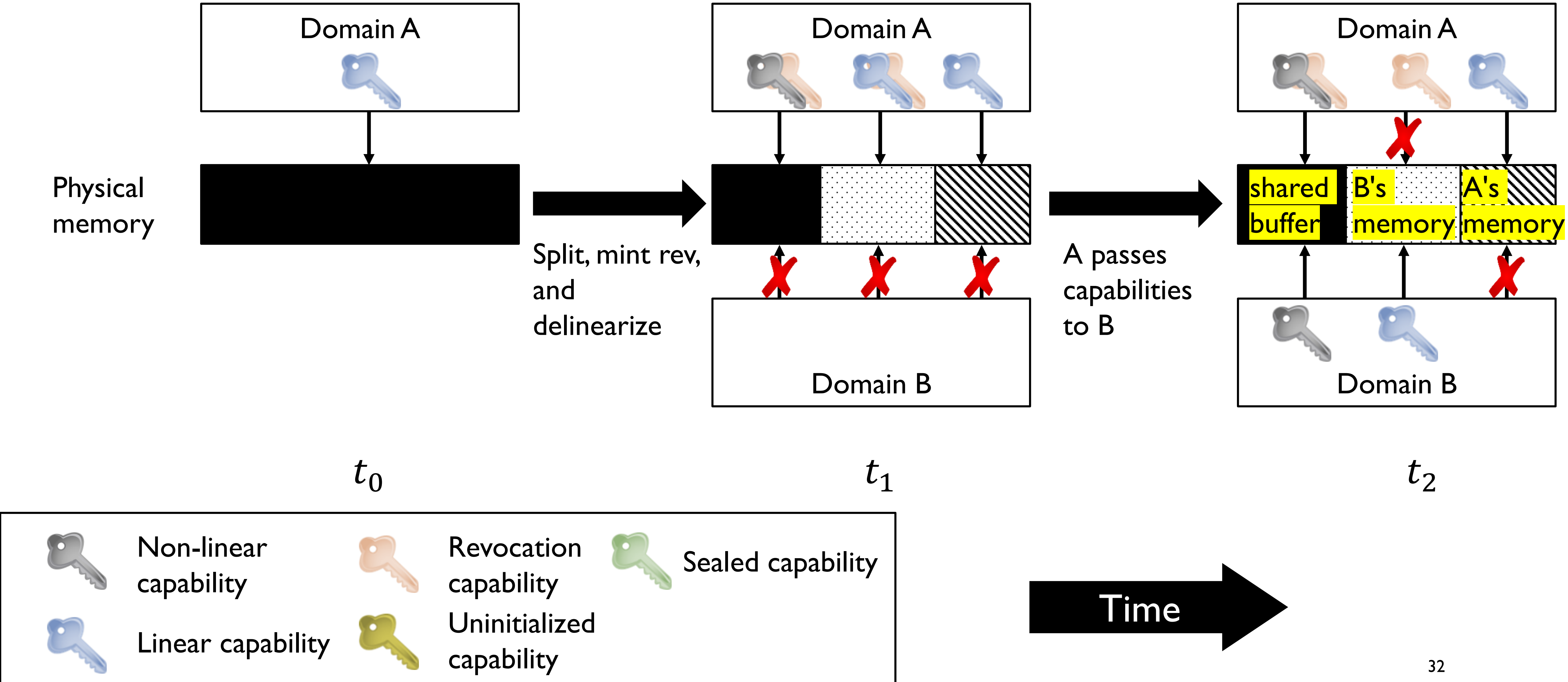
Trustless Memory Allocator



Trustless Scheduler



Nestable Enclaves



Domain A

Physical memory

Split, mint rev, and delinearize

Domain A

Domain B

A passes capabilities to B

Domain A

shared buffer

B's memory

A's memory

Domain B

t_0

t_1

t_2

Non-linear capability

Linear capability

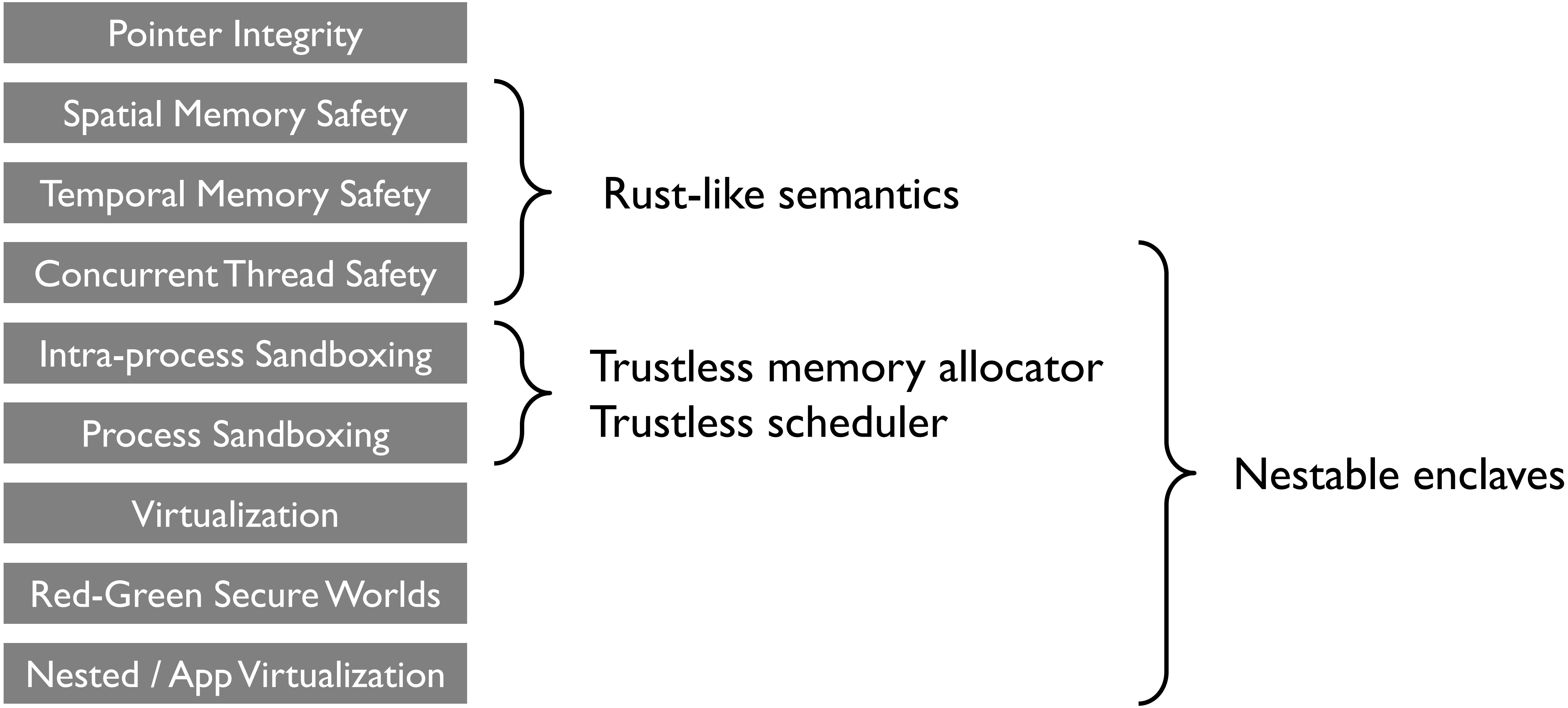
Revocation capability

Uninitialized capability

Sealed capability

Time

Case Studies



Pointer Integrity

Spatial Memory Safety

Temporal Memory Safety

Concurrent Thread Safety

Intra-process Sandboxing

Process Sandboxing

Virtualization

Red-Green Secure Worlds

Nested / App Virtualization

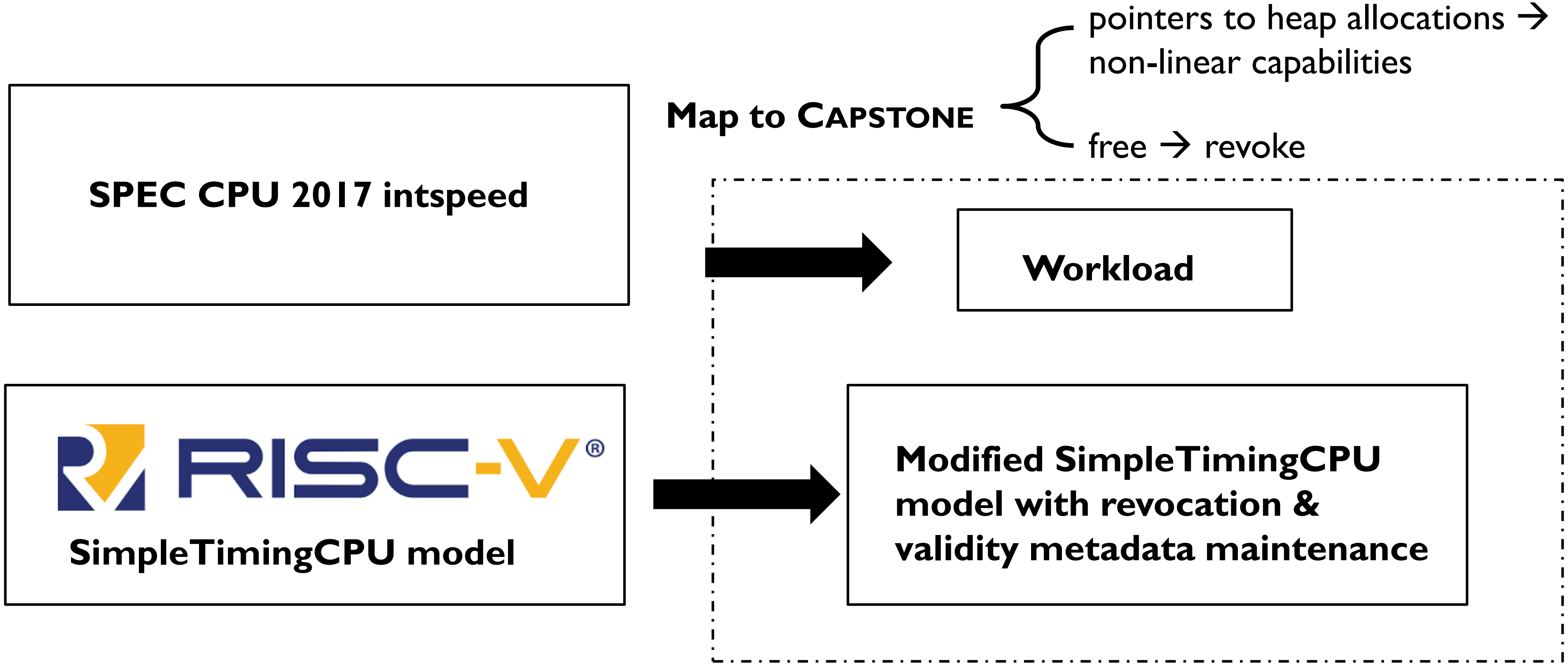
Rust-like semantics

Trustless memory allocator
Trustless scheduler

Nestable enclaves

Takeaway: CAPSTONE is highly expressive

Preliminary Performance Evaluation



SPEC CPU 2017 intspeed

SimpleTimingCPU model

Map to CAPSTONE

pointers to heap allocations → non-linear capabilities

free → revoke

Workload

Modified SimpleTimingCPU model with revocation & validity metadata maintenance

Results: within ~50% run time overhead

Conclusion

- **Goal: unified foundation for trustless memory access**
- **Required properties**
  - **Exclusive access**
  - **Revocable delegation**
  - **Extensible hierarchy**
  - **Secure domain switching**
- **CAPSTONE**
  - **Capability-based architecture**
- Core ideas: linear capabilities, revocation, uninitialized capabilities
- Prototype implementations with emulator, compiler, and library
- Case studies: CAPSTONE is highly expressive

https://capstone.kisp-lab.org/

Thanks for listening!