

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

32<sup>nd</sup> USENIX Security Symposium

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

National University of Singapore  
University of Cambridge\*



# World of Security Extensions

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

# Problems with Security Extensions

## I. 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:

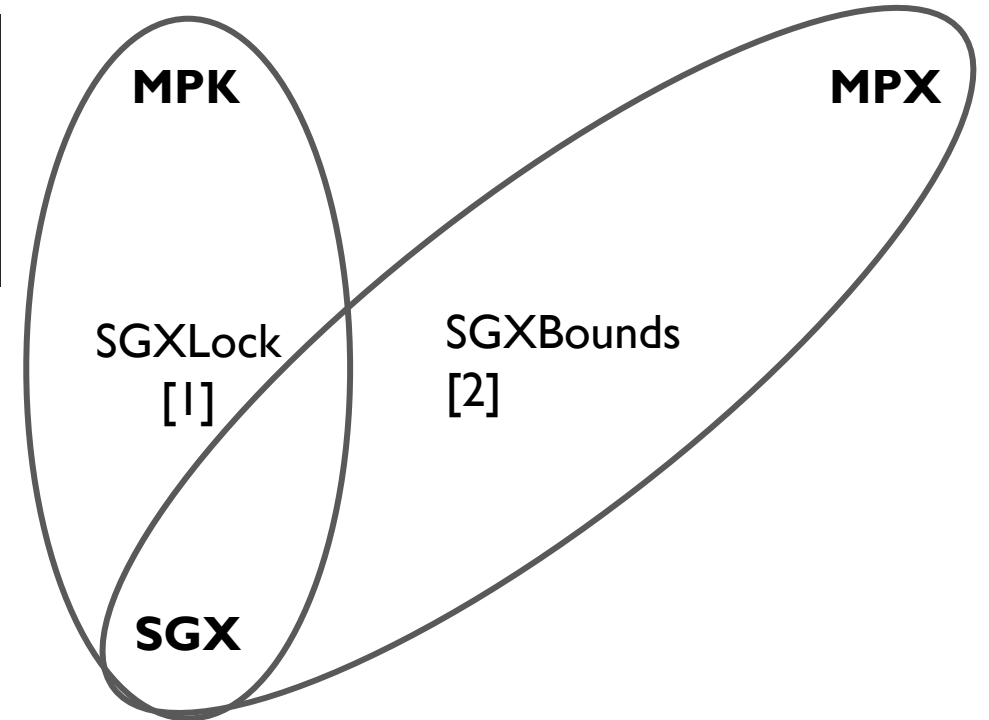
- Intel® Memory Protection Extensions (Intel® MPX)
- Branch Monitoring Counters
- Hardware Lock Elision (HLE), part of Intel® TSX-NI
- Intel® Software Guard Extensions (Intel® SGX)
- Intel® TSX-NI
- Power Aware Interrupt Routing (PAIR)

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

[1] Y. Chen et al., 'SGXLock: Towards Efficiently Establishing Mutual Distrust Between Host Application and Enclave for SGX', in *31st USENIX Security Symposium, 2022*

[2] D. Kuvaiskii et al., 'SGXBOUNDS: Memory Safety for Shielded Execution', in *Proceedings of the Twelfth European Conference on Computer Systems*

## 2. Poor interoperability for multiple security goals



# Problems with Security Extensions

## 1. Unreliable availability of security features

## 2. Poor interoperability for multiple security goals

```
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

**Is there a unified foundation for multiple security goals?**

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

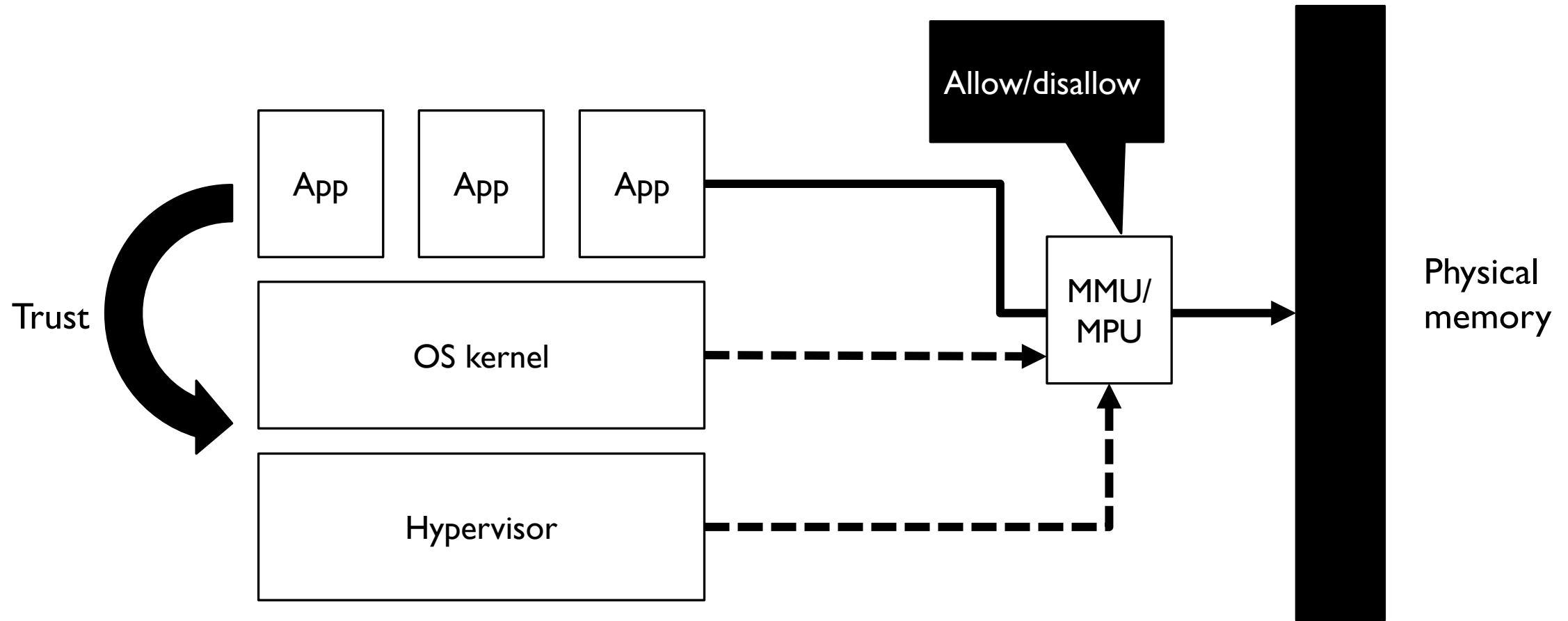
- Intel® Memory Protection Extensions (Intel® MPX)
- Branch Monitoring Counters
- Hardware Lock Elision (HLE), part of Intel® TSX-NI
- Intel® Software Guard Extensions (Intel® SGX)
- Intel® TSX-NI
- Power Aware Interrupt Routing (PAIR)

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

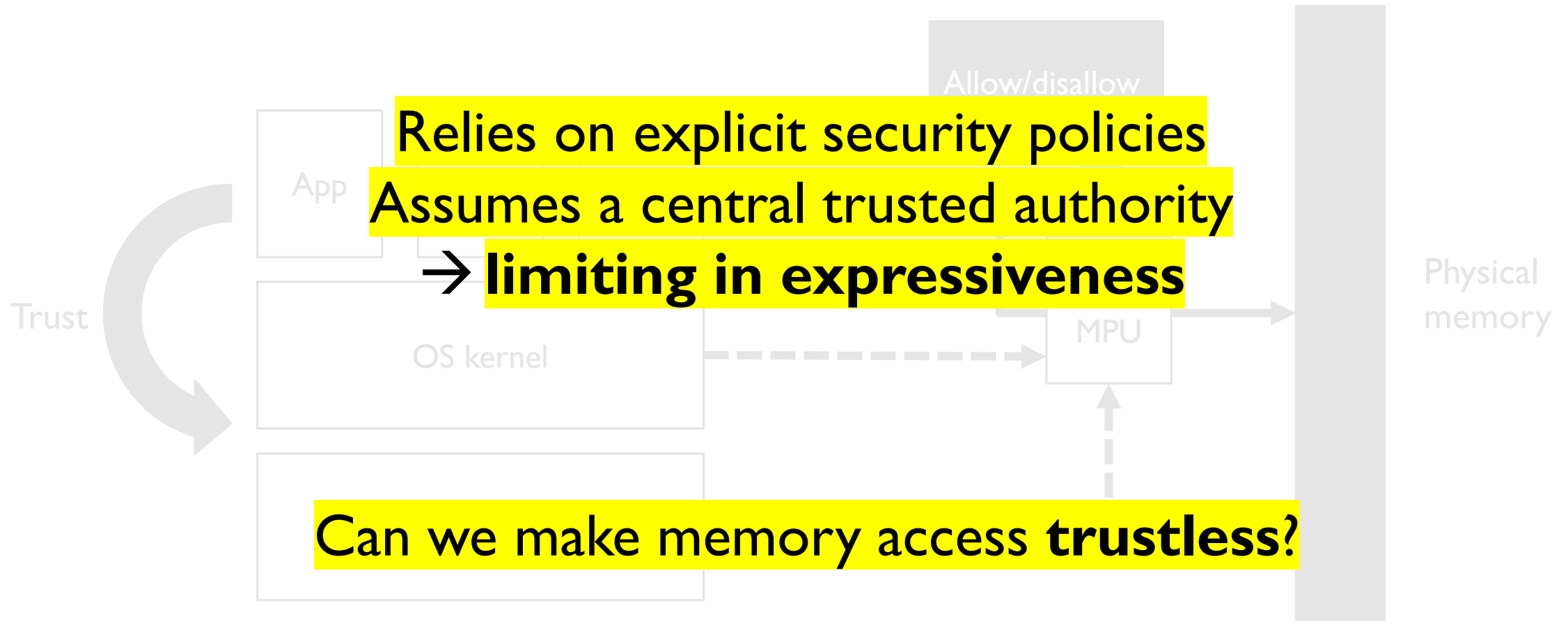
[1] Y. Chen et al., 'SGXLock: Towards Efficiently Establishing Mutual Distrust Between Host Application and Enclave for SGX', in *31st USENIX Security Symposium, 2022*

[2] D. Kuvaiskii et al., 'SGXBOUNDS: Memory Safety for Shielded Execution', in *Proceedings of the Twelfth European Conference on Computer Systems*

# Traditional Architectures Rely on Access Control



# Traditional Architectures Rely on Access Control



# Contributions

## 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

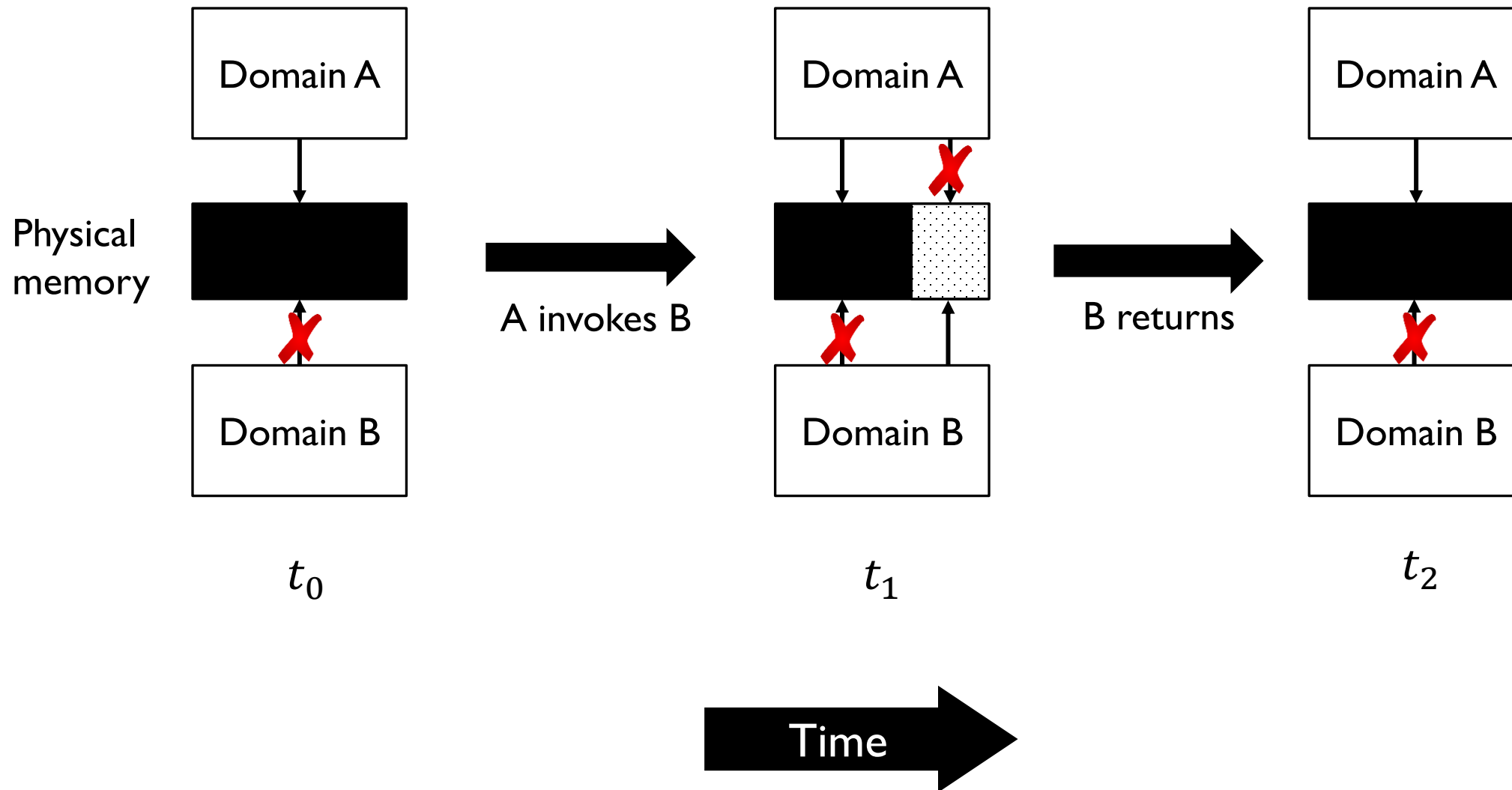
Process Sandboxing

Virtualization

Red-Green Secure Worlds

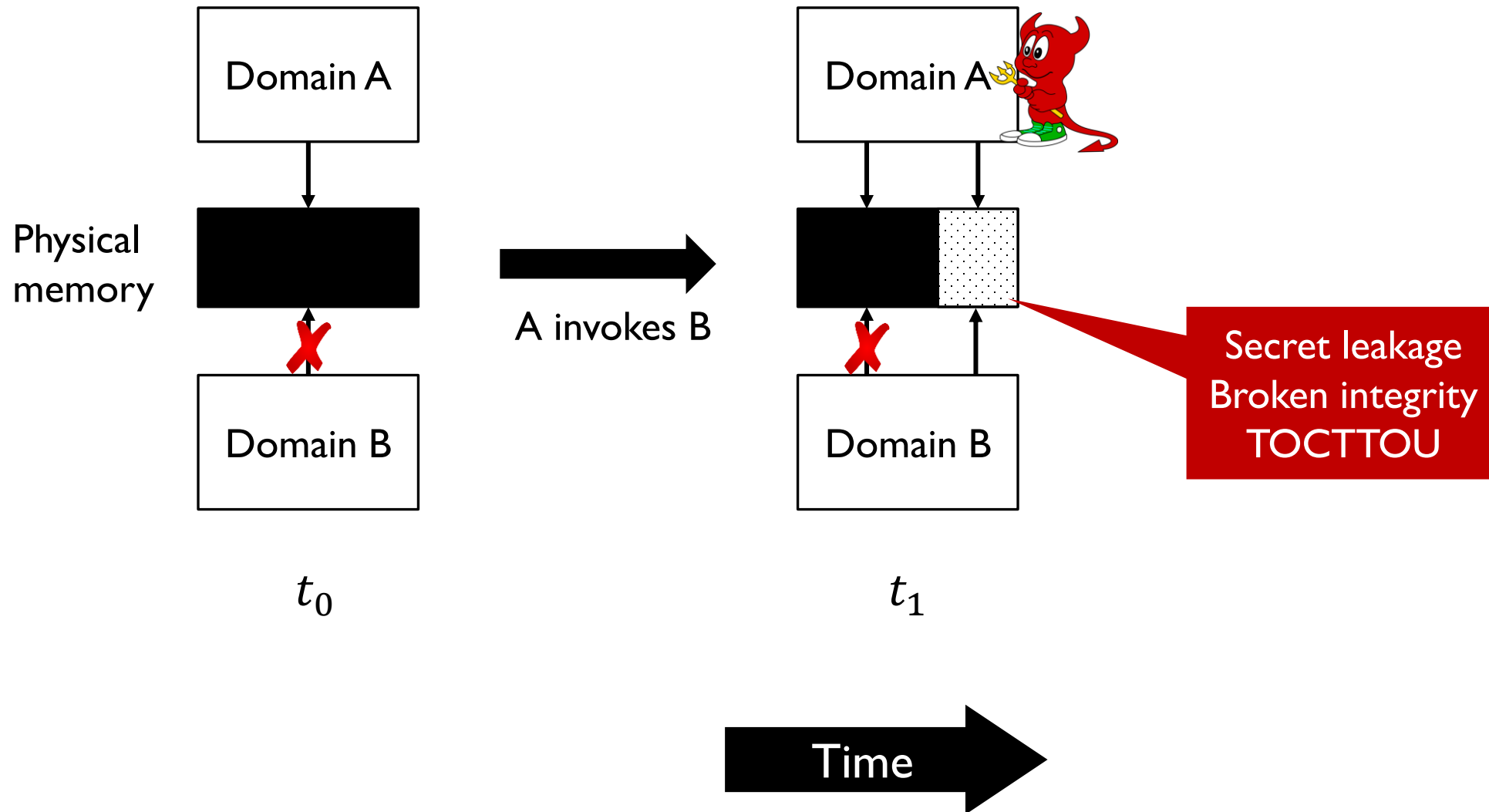
Nested / App Virtualization

# Threat Model: Benign Scenario

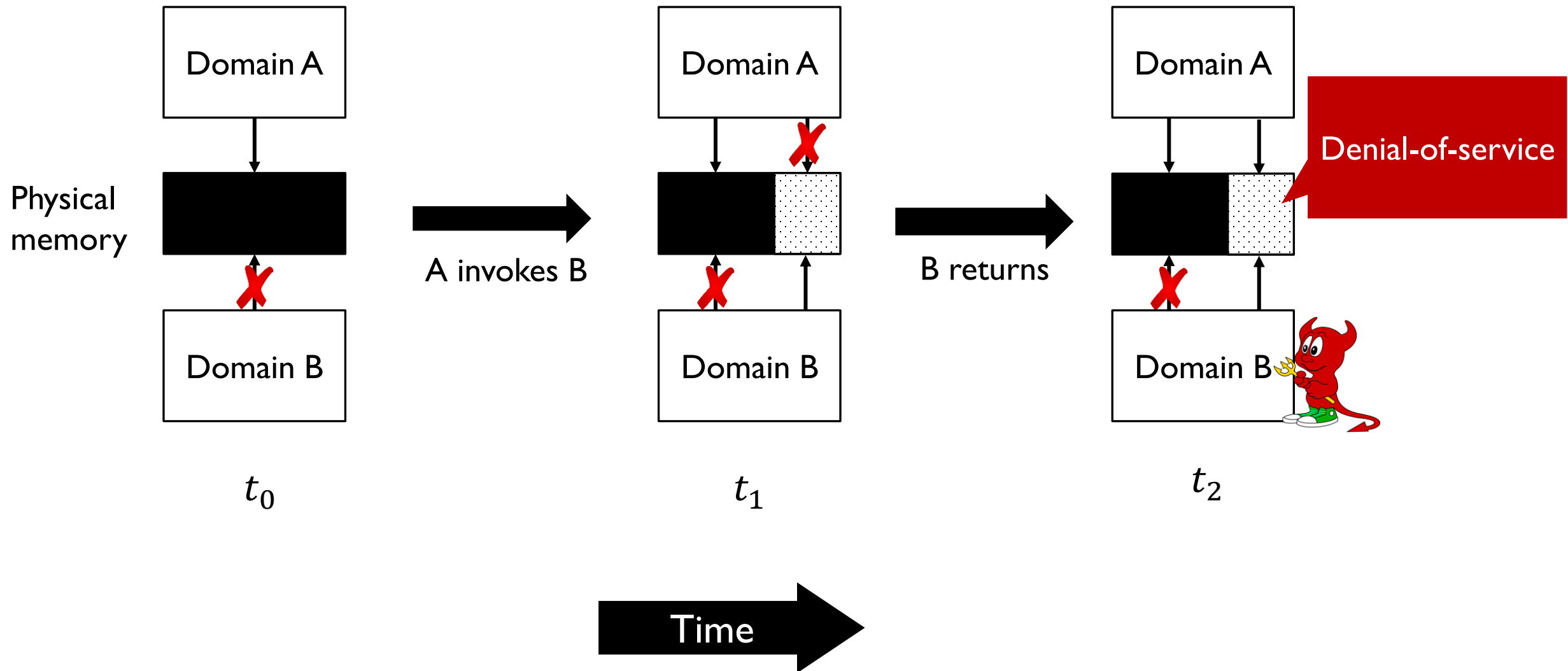




# Threat Model: Malicious Scenario



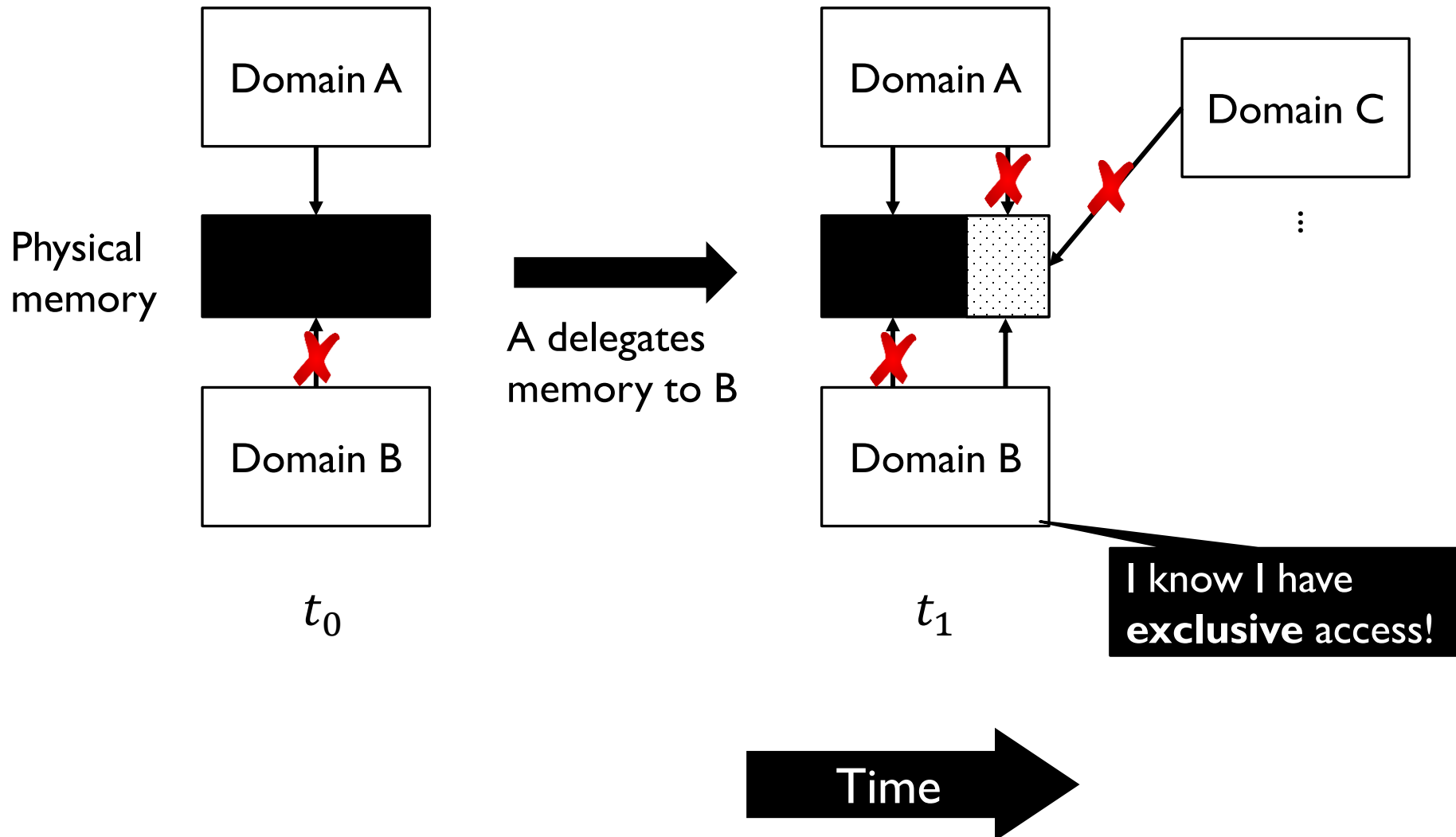
# Threat Model: Malicious Scenario



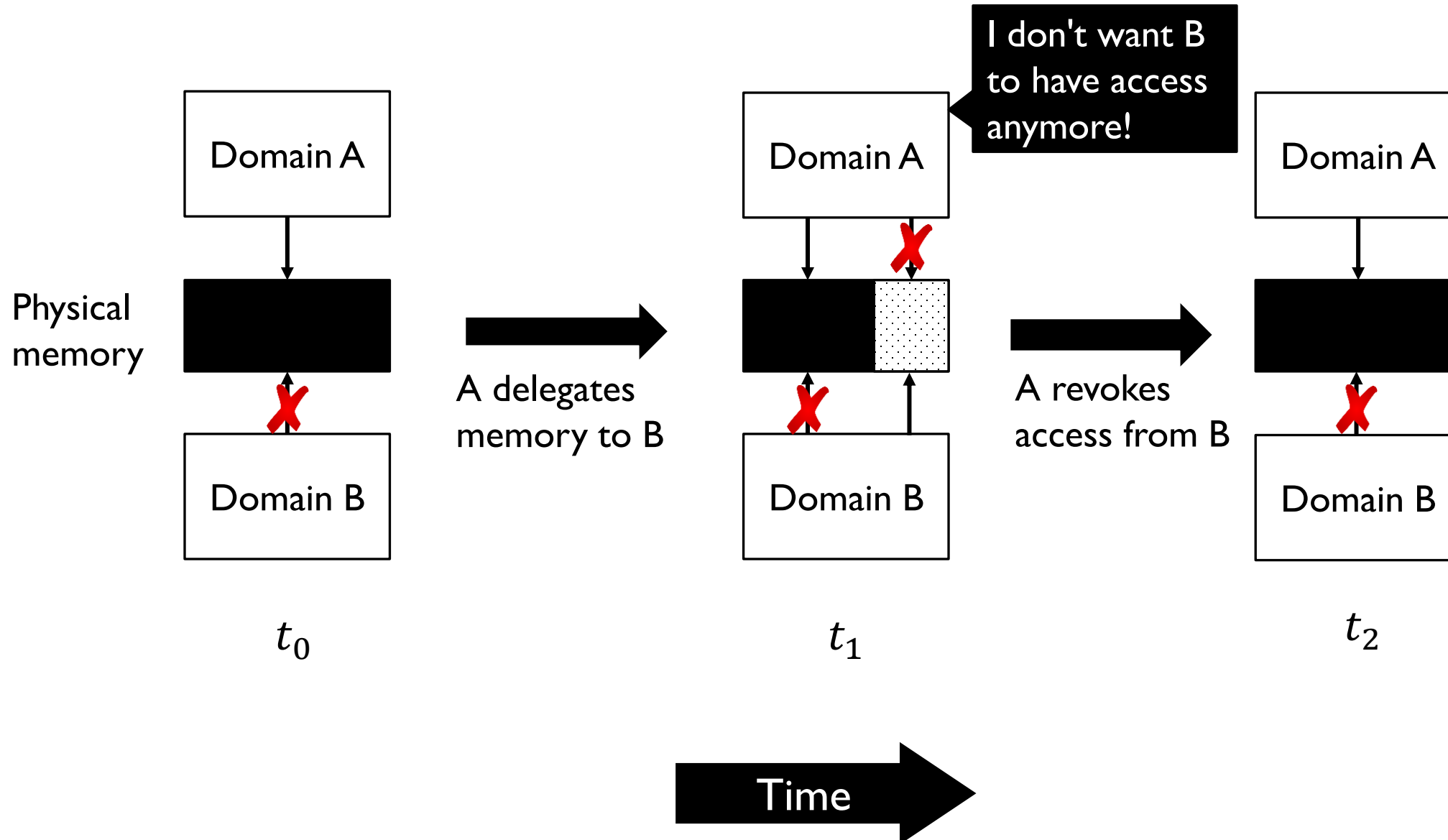
# Minimal set of properties for a unified foundation

---

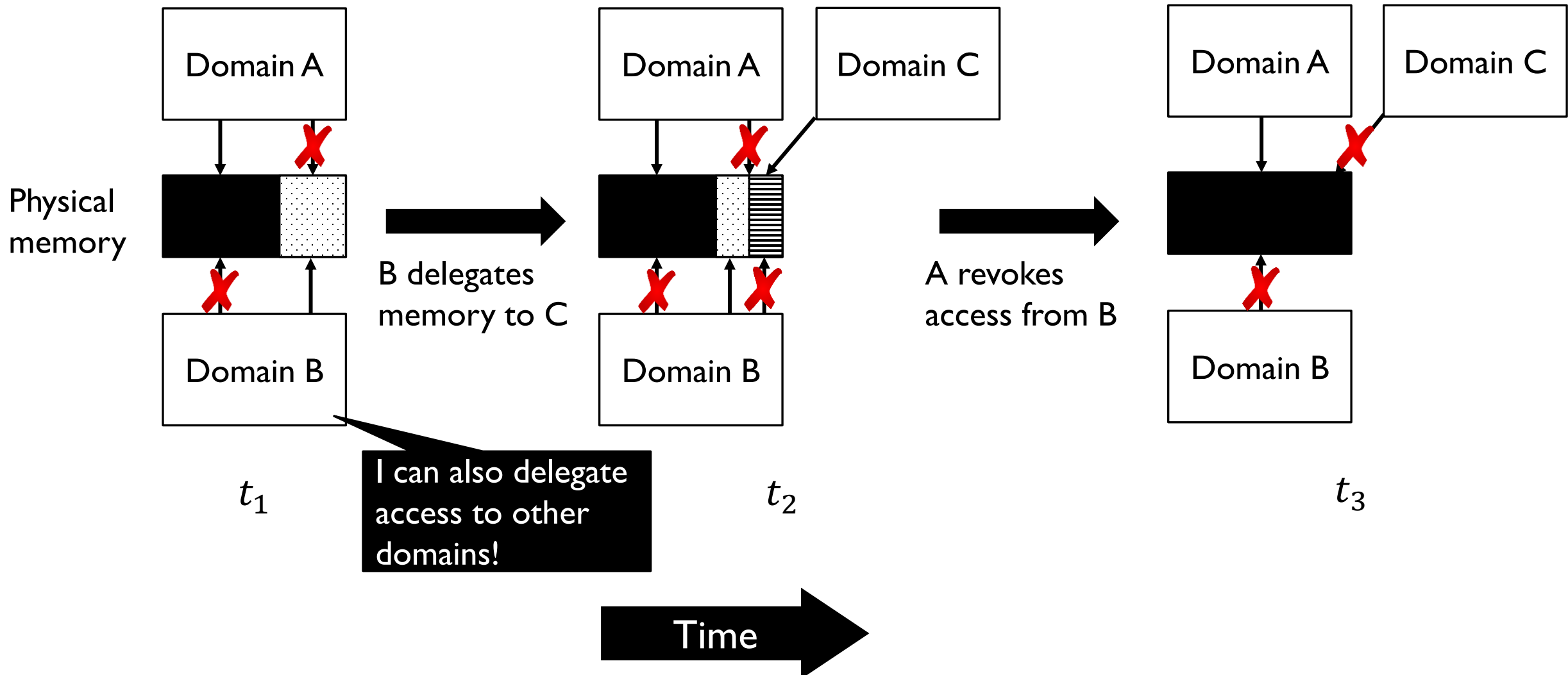
# Property I: Exclusive Access



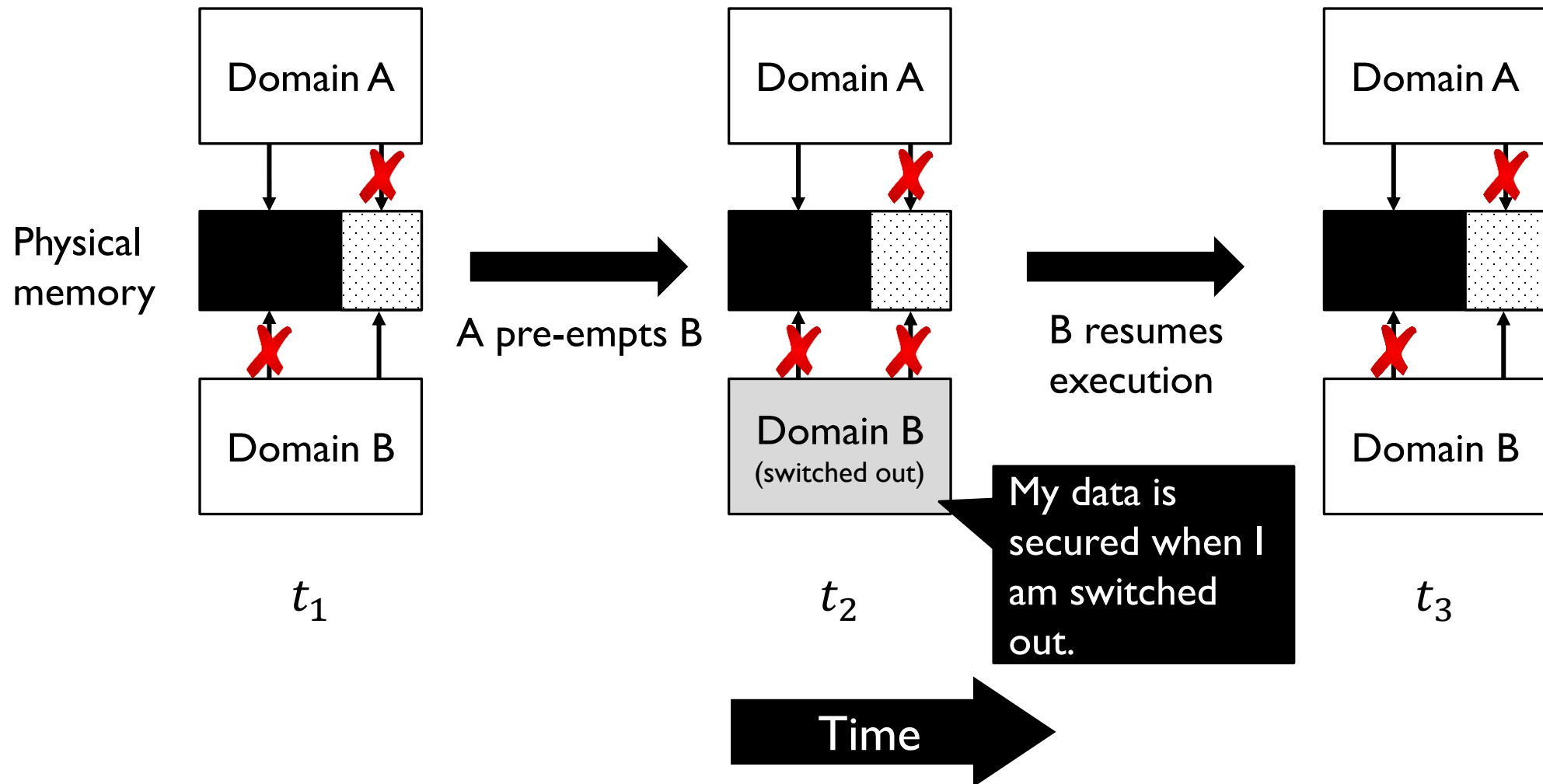
# Property 2: Revocable Delegation



# Property 3: Extensible Hierarchy



# Property 4: Secure Domain Switching



# Properties for a Trustless Unified Foundation

***P1: Exclusive Access***

***P2: Revocable Delegation***

***P3: Extensible Hierarchy***

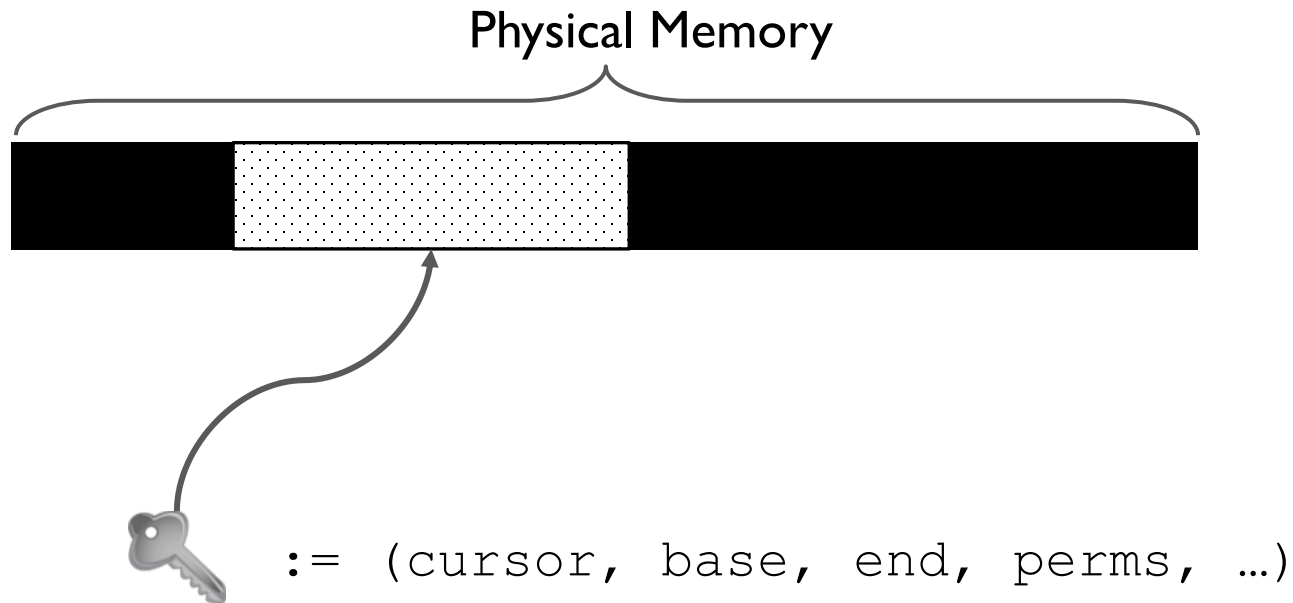
***P4: Secure Domain Switching***

How to enforce those properties through a unified interface?





# Architectural Capabilities: A Baseline

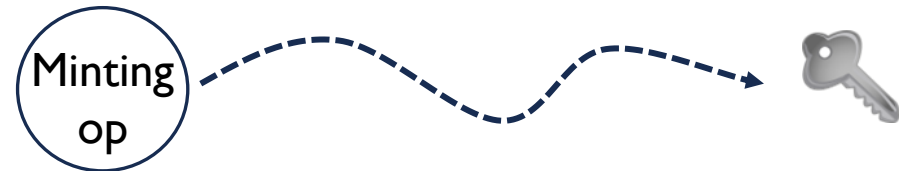


Capability

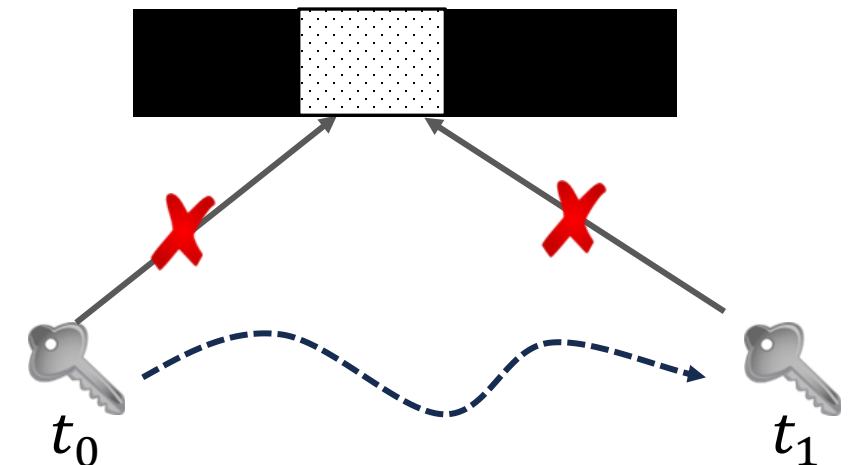
LD/ST ~~addr~~, ...

LD/ST , ...

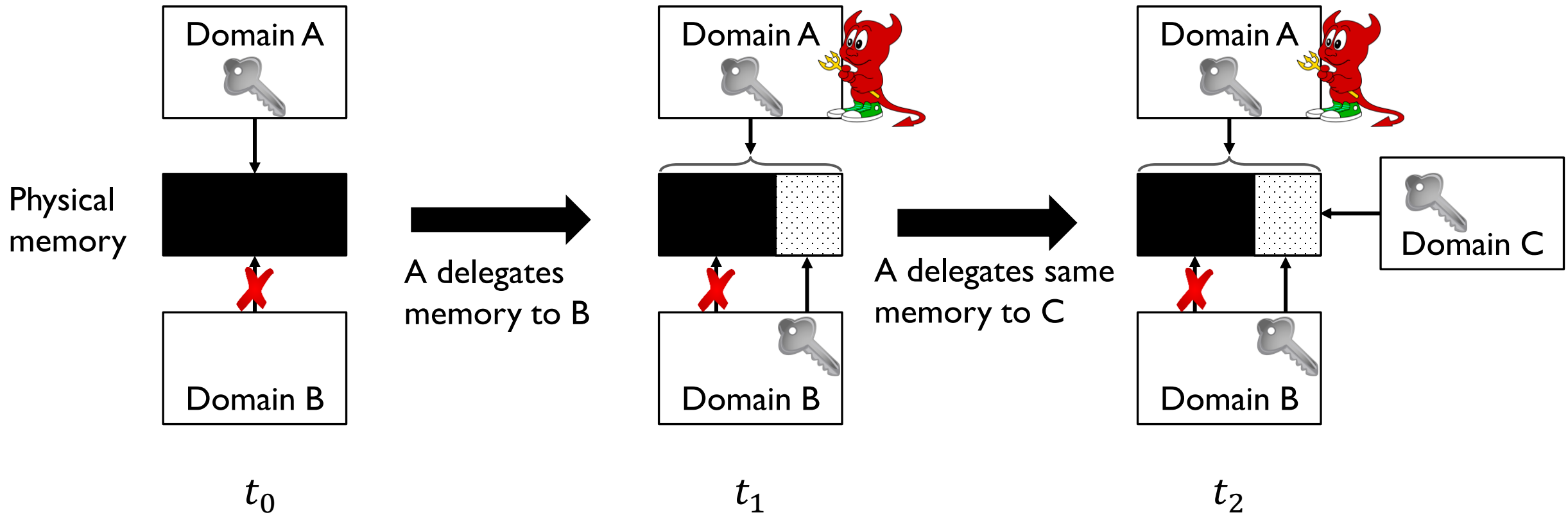
Unforgeability



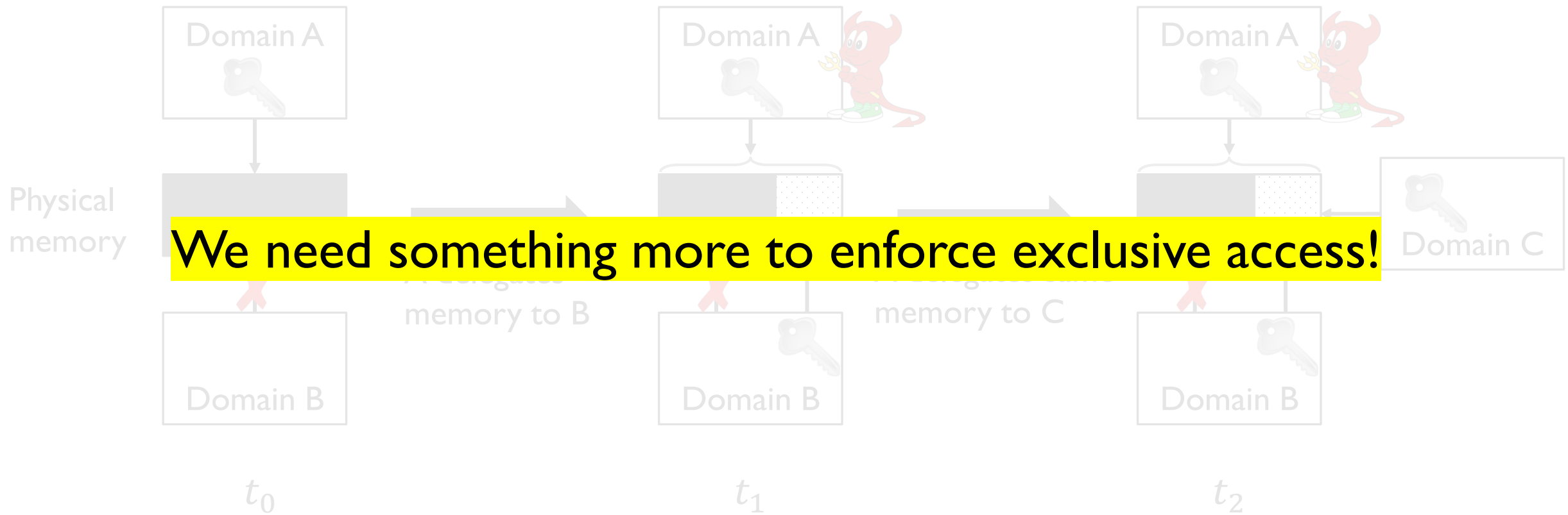
Monotonicity



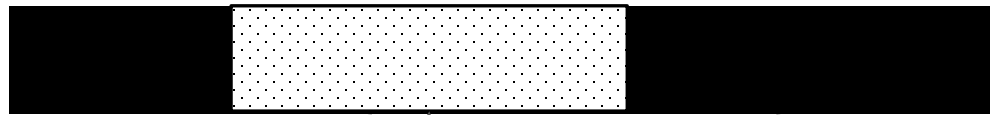
# Enforcing Property I: Exclusive Access



# Enforcing Property 1: Exclusive Access



# Exclusive Access: Linear Capabilities

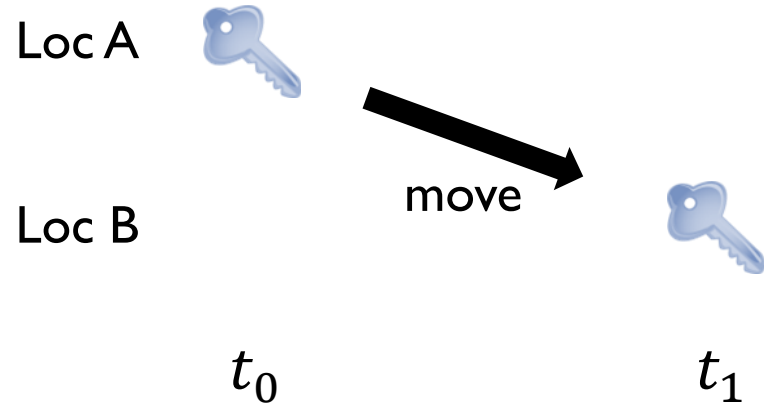


Linear  
capability

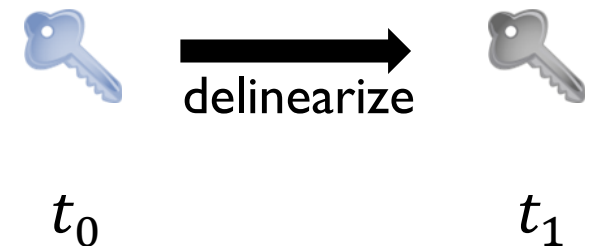
✓ Exclusive access

## Linear Capability Operations

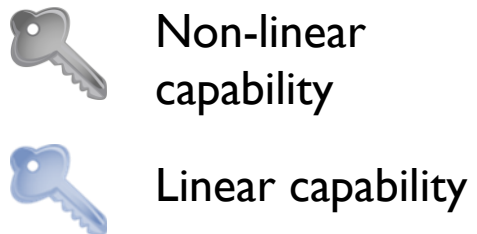
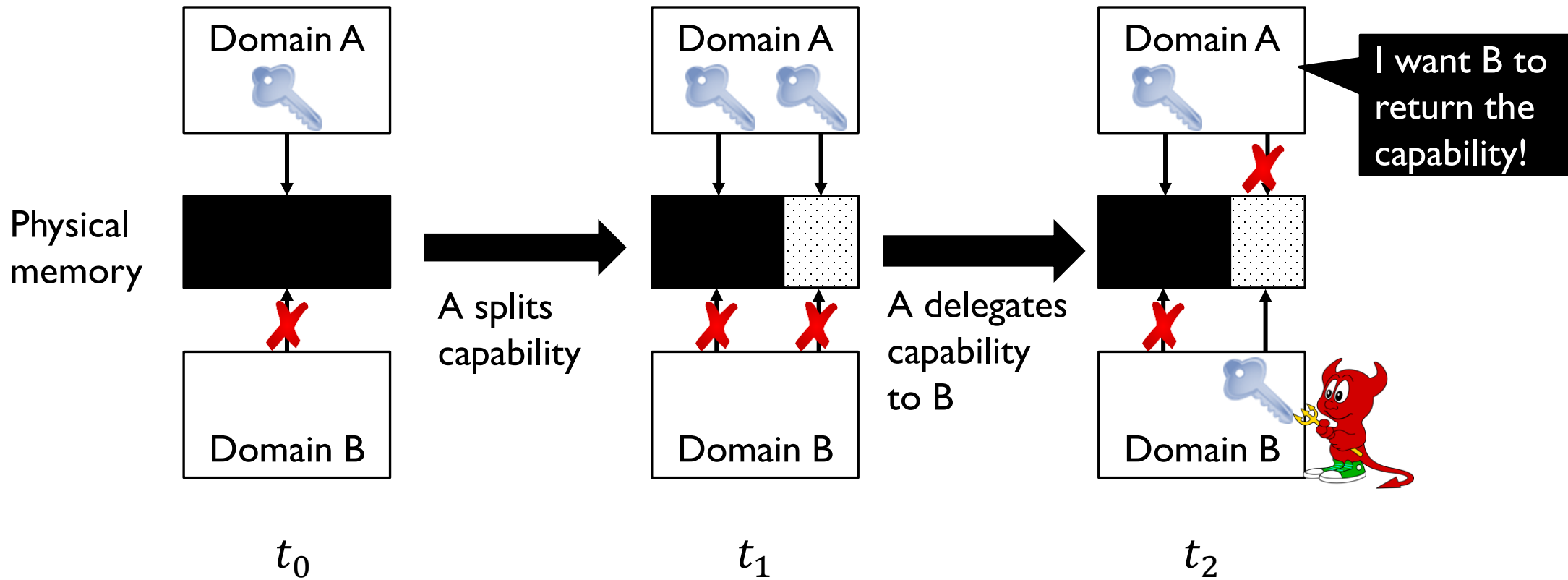
Move



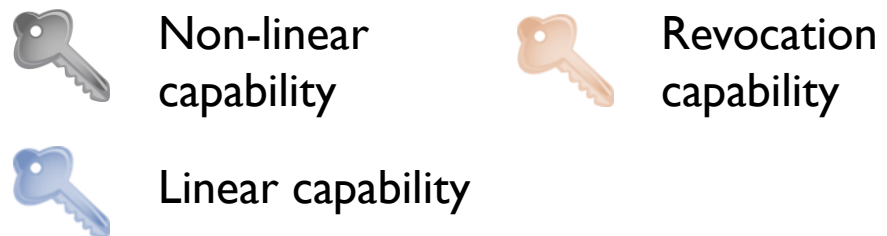
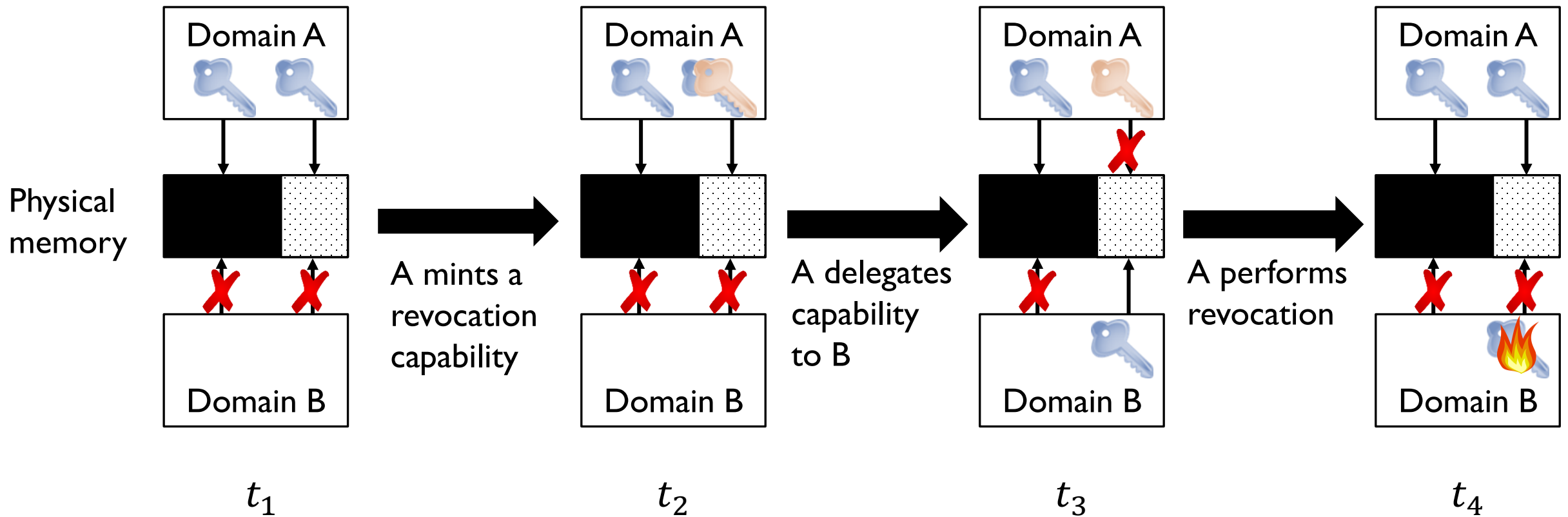
Delinearize



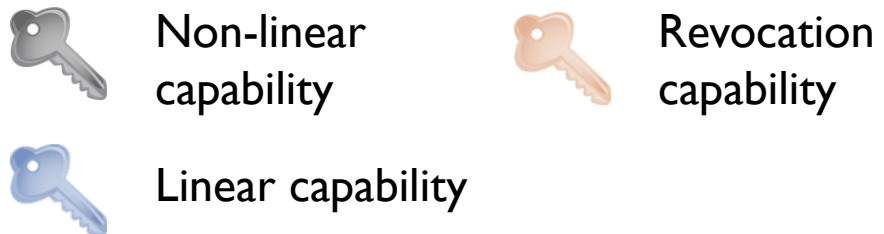
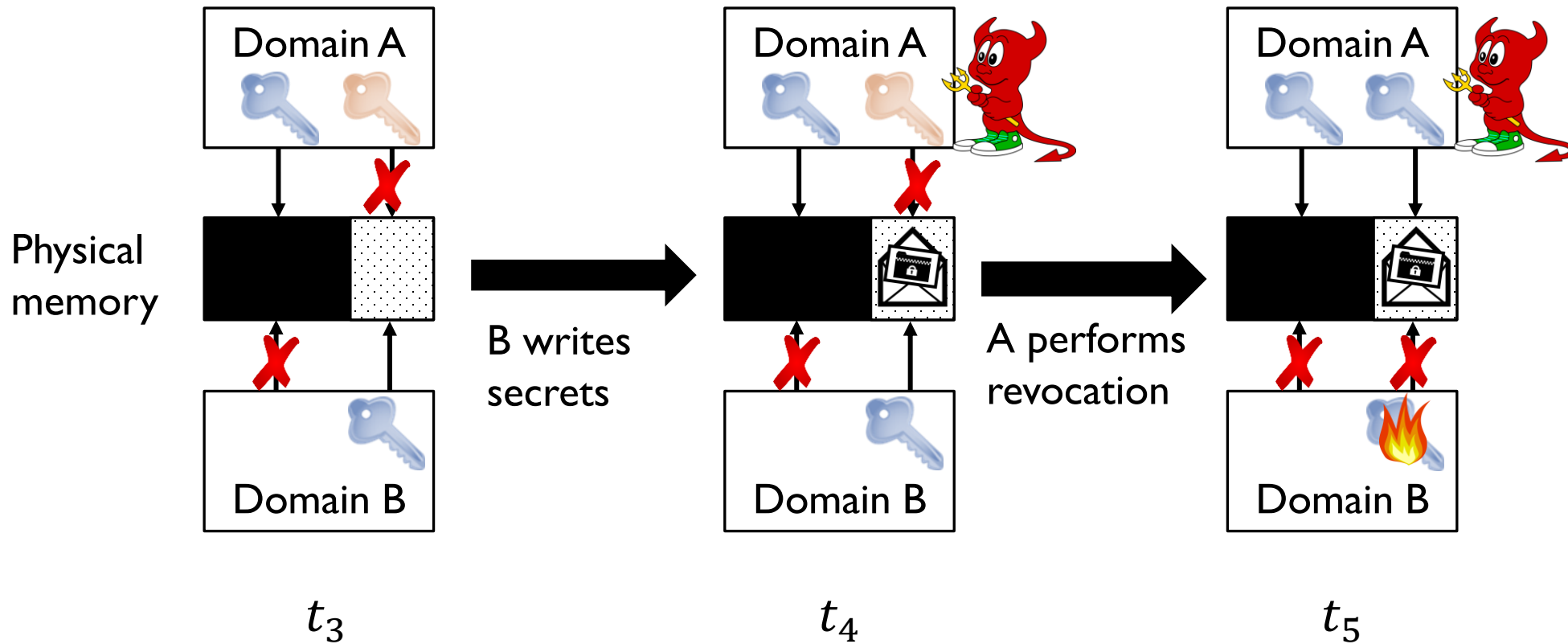
# Memory Delegation with Linear Capabilities



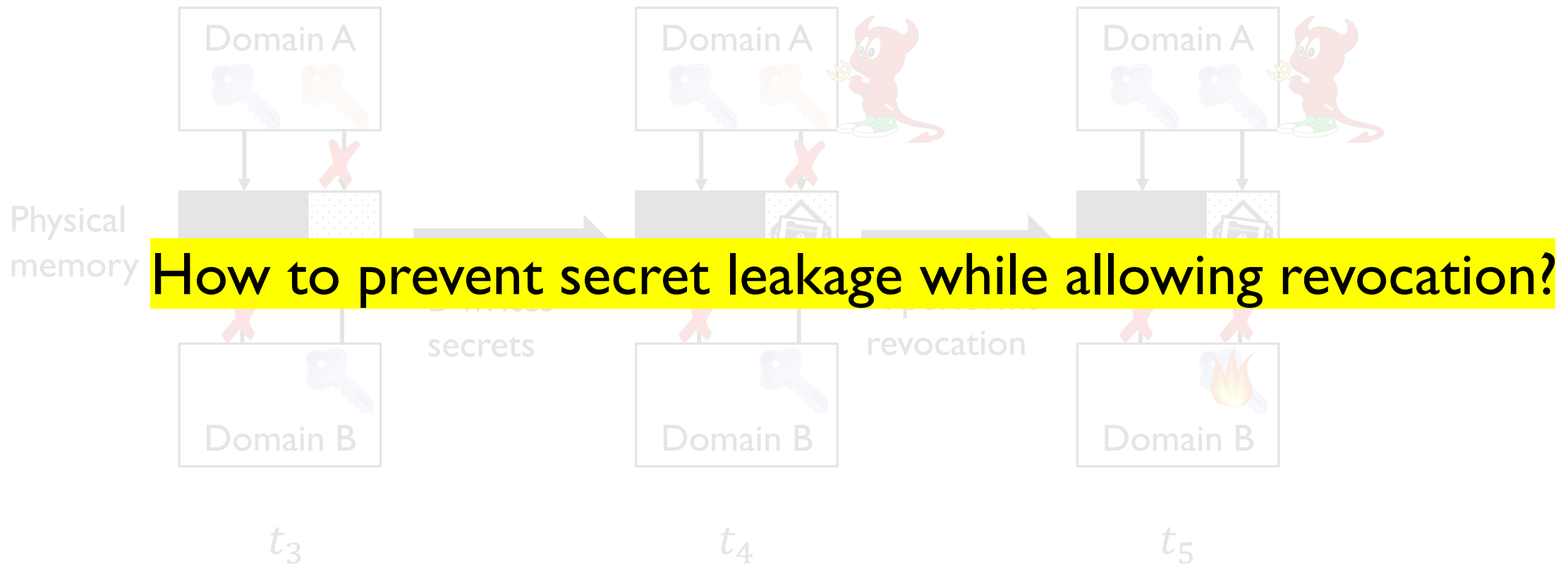
# Enforcing Property 2: Revocable Delegation



# Problem: Secret Leakage Can Happen



# Problem: Secret Leakage Can Happen



Non-linear capability



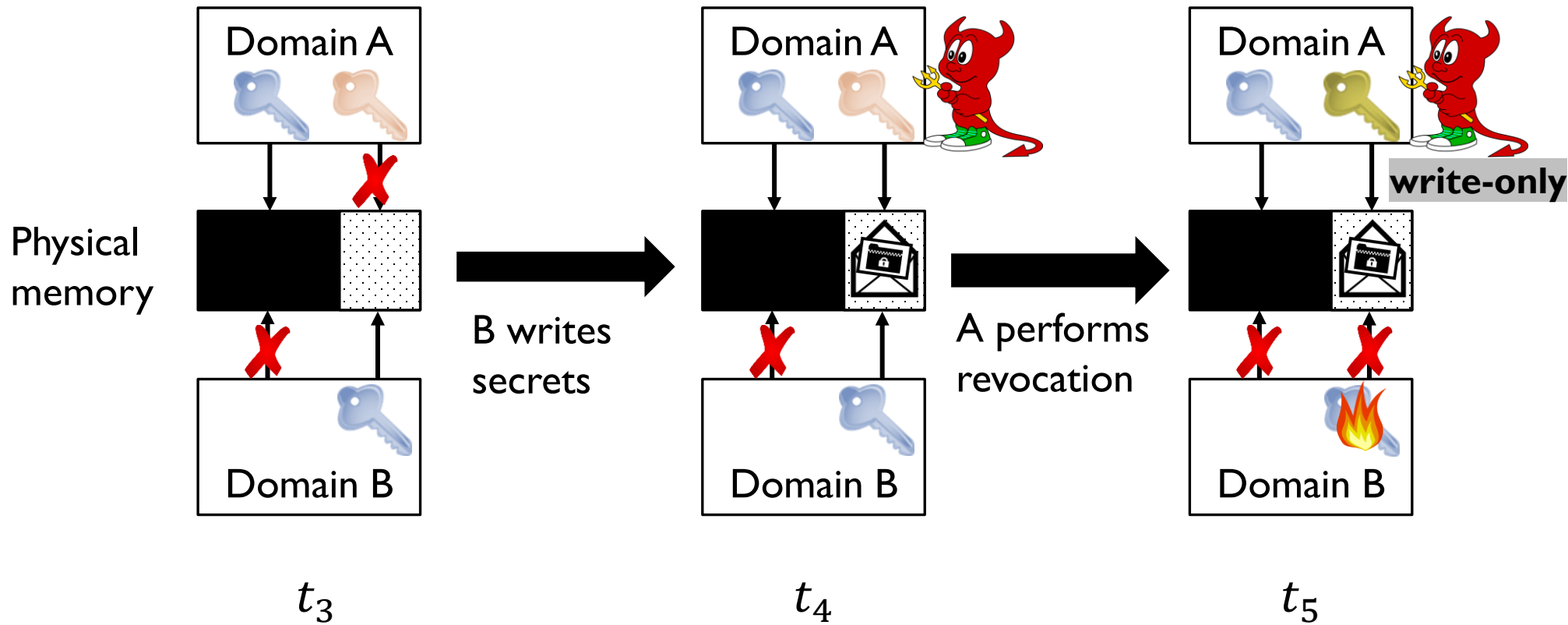
Revocation capability



Linear capability



# Solution: Uninitialized Capabilities



# Properties for a Trustless Unified Foundation

***P1: Exclusive Access***

***P2: Revocable Delegation***

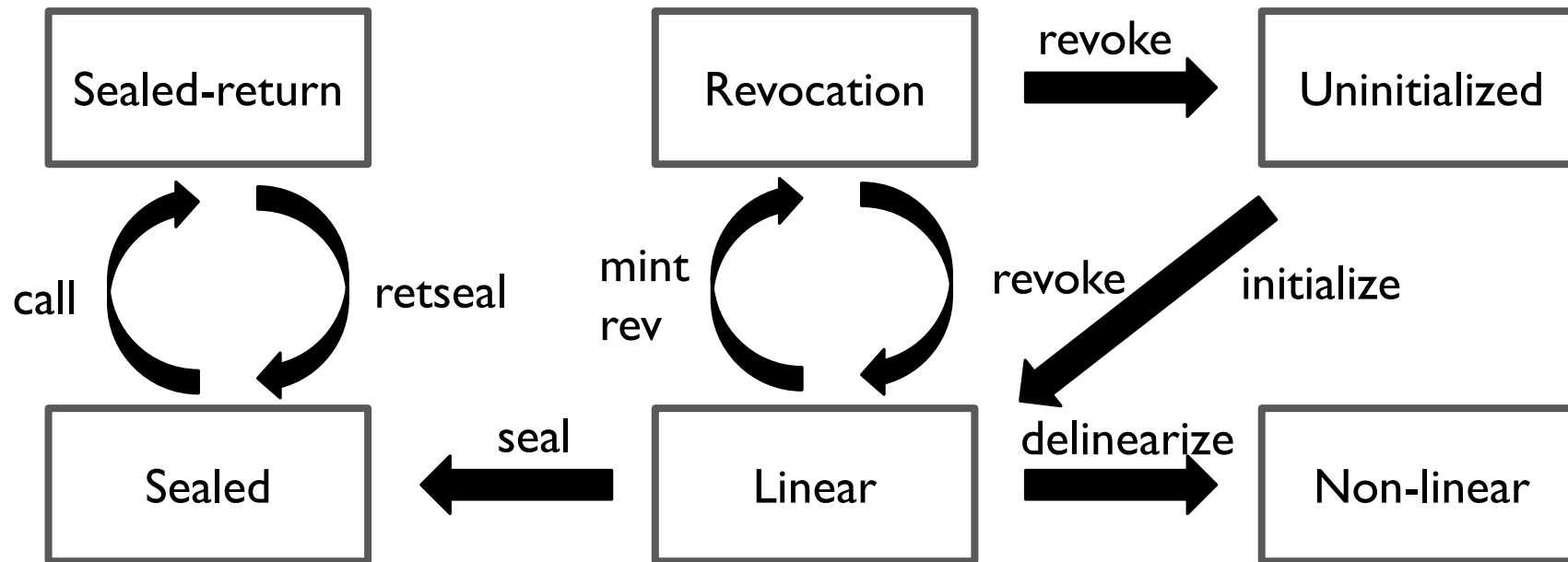
***P3: Extensible Hierarchy***

***P4: Secure Domain Switching***

**Please see paper!**

# CAPSTONE: Putting It Together

ISA with capability types and instructions

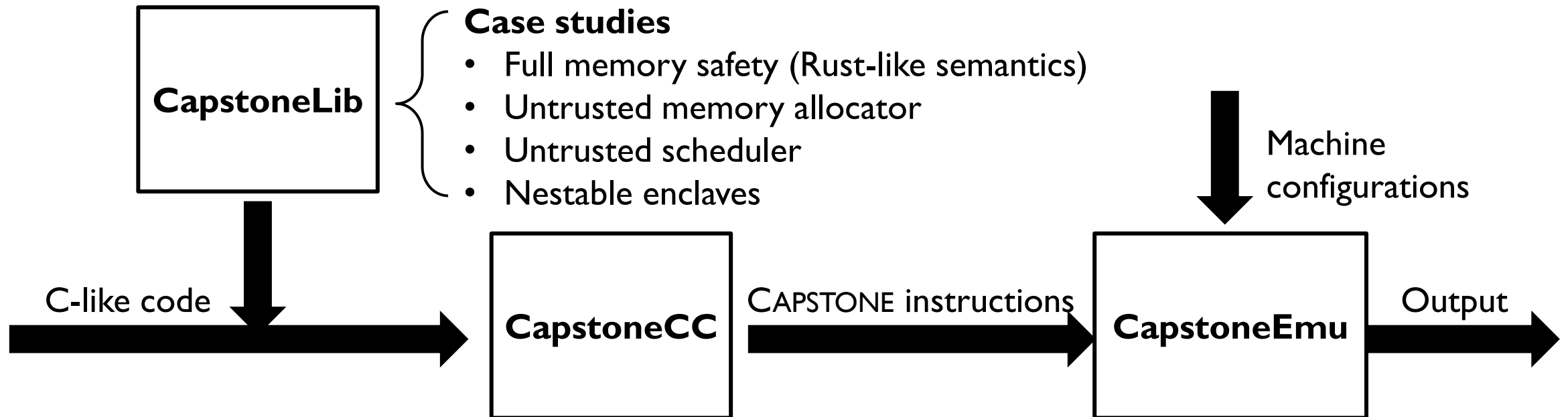


<https://capstone.kisp-lab.org/>

# Implementation and Evaluation

---

# Functional Prototype



# Case Study: Memory Safety (Rust-like Semantics)

Spatial Memory Safety

Temporal Memory Safety

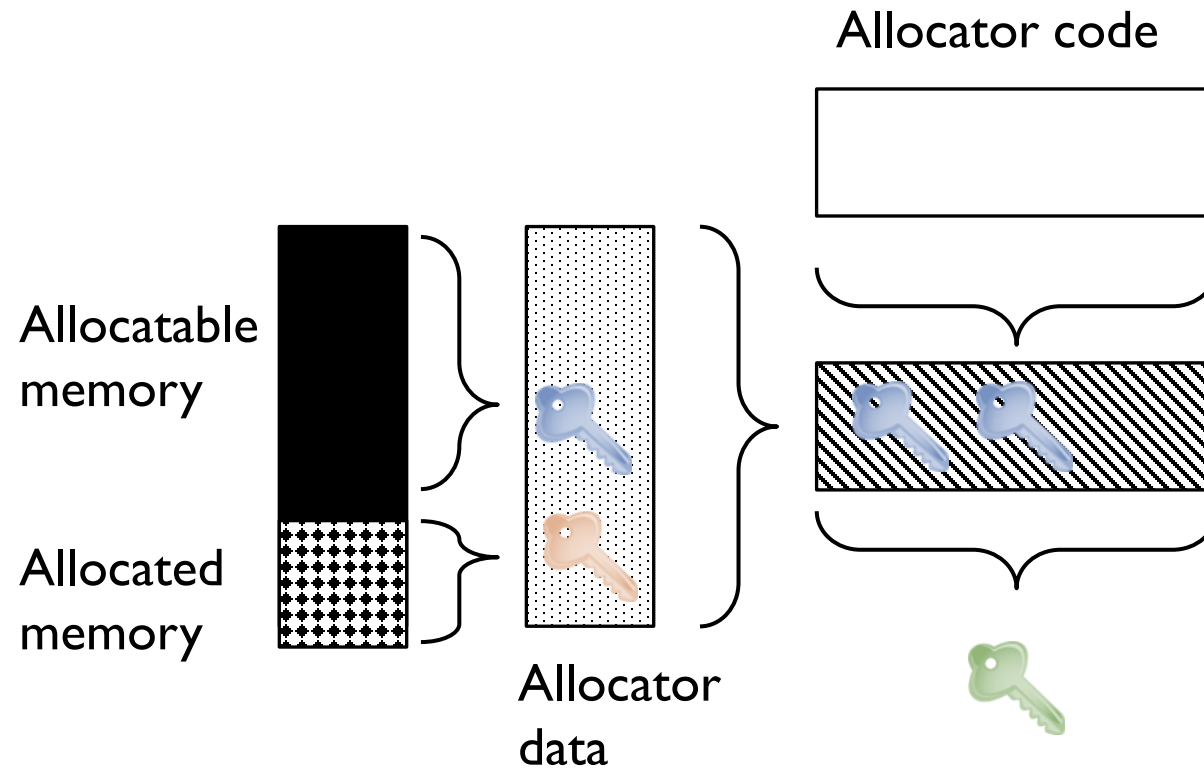
Concurrent Thread Safety






} Architectural capabilities

} Linear capabilities + revocation

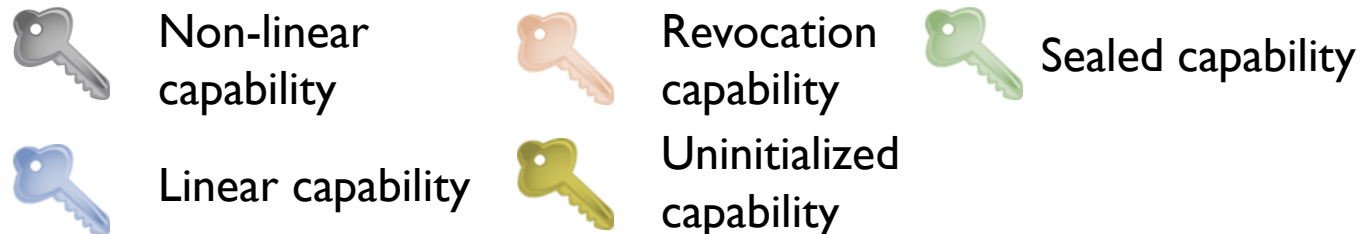
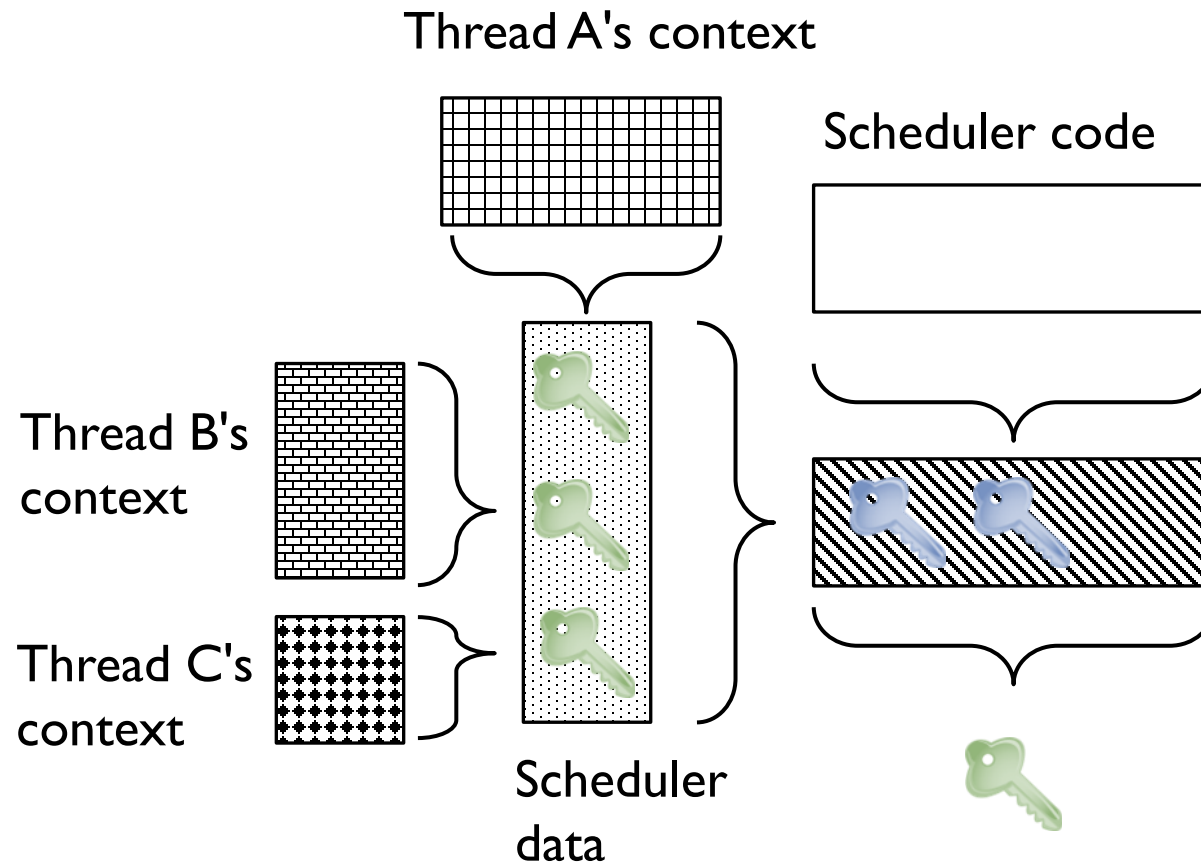
Operation	Rust semantics	CAPSTONE
Move	<code>let a = b;</code>	<code>mov ra, rb;</code>
Immutable borrow	<code>let a = &amp;b;</code>	<code>mrev rr, rb; delin rb; li r0, 0; tighten rb, r0; mov ra, rb; (use ra) revoke rr; mov rb, rr</code>
Mutable borrow	<code>let a = &amp;mut b;</code>	<code>mrev rr, rb; mov ra, rb; (use ra) revoke rr; mov rb, rr</code>

# Case Study: Trustless Memory Allocator



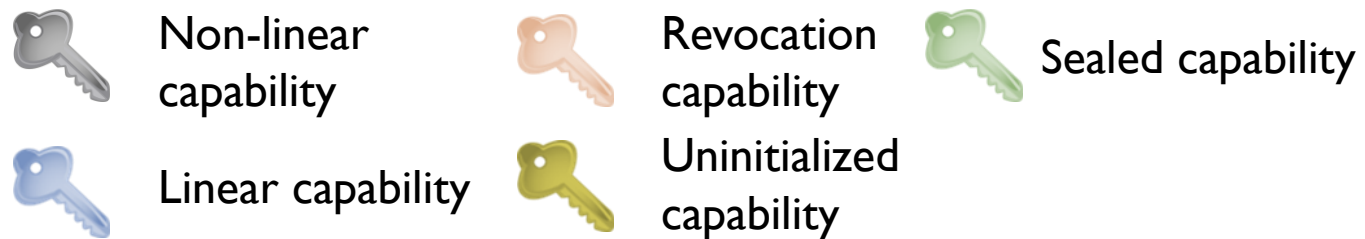
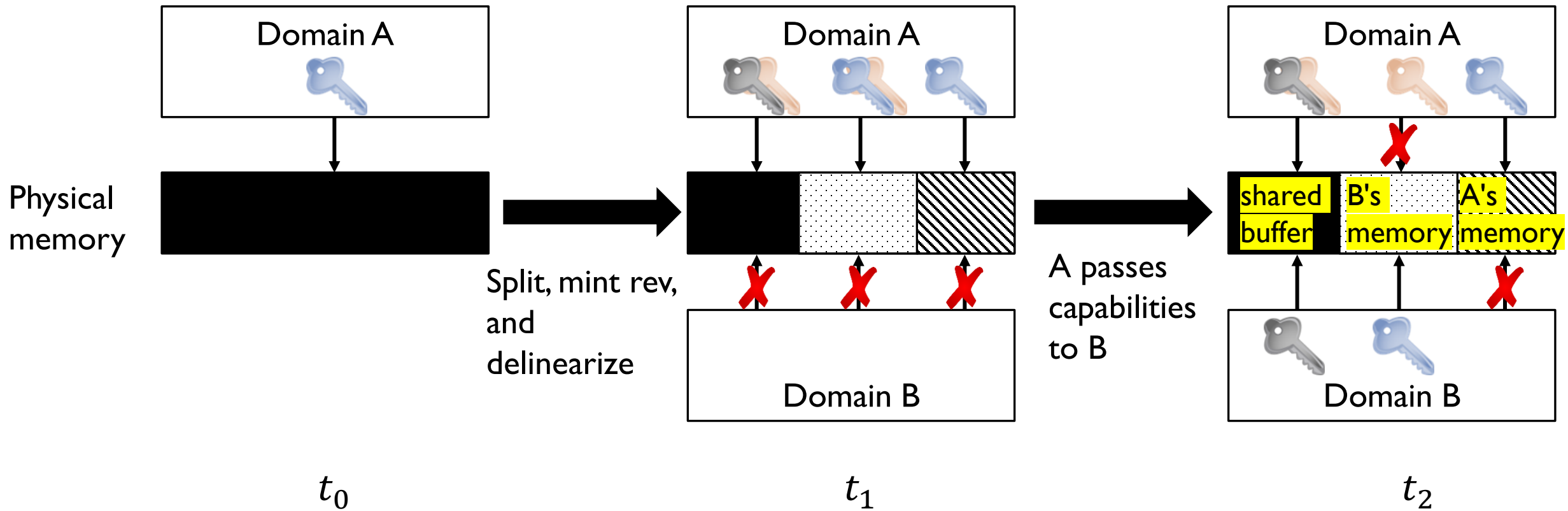
- |  |                       |   |                          |  |                   |
|--|-----------------------|---|--------------------------|--|-------------------|
|  | Non-linear capability |  | Revocation capability    |  | Sealed capability |
|  | Linear capability     |  | Uninitialized capability |  |                   |

# Case Study: Trustless Scheduler

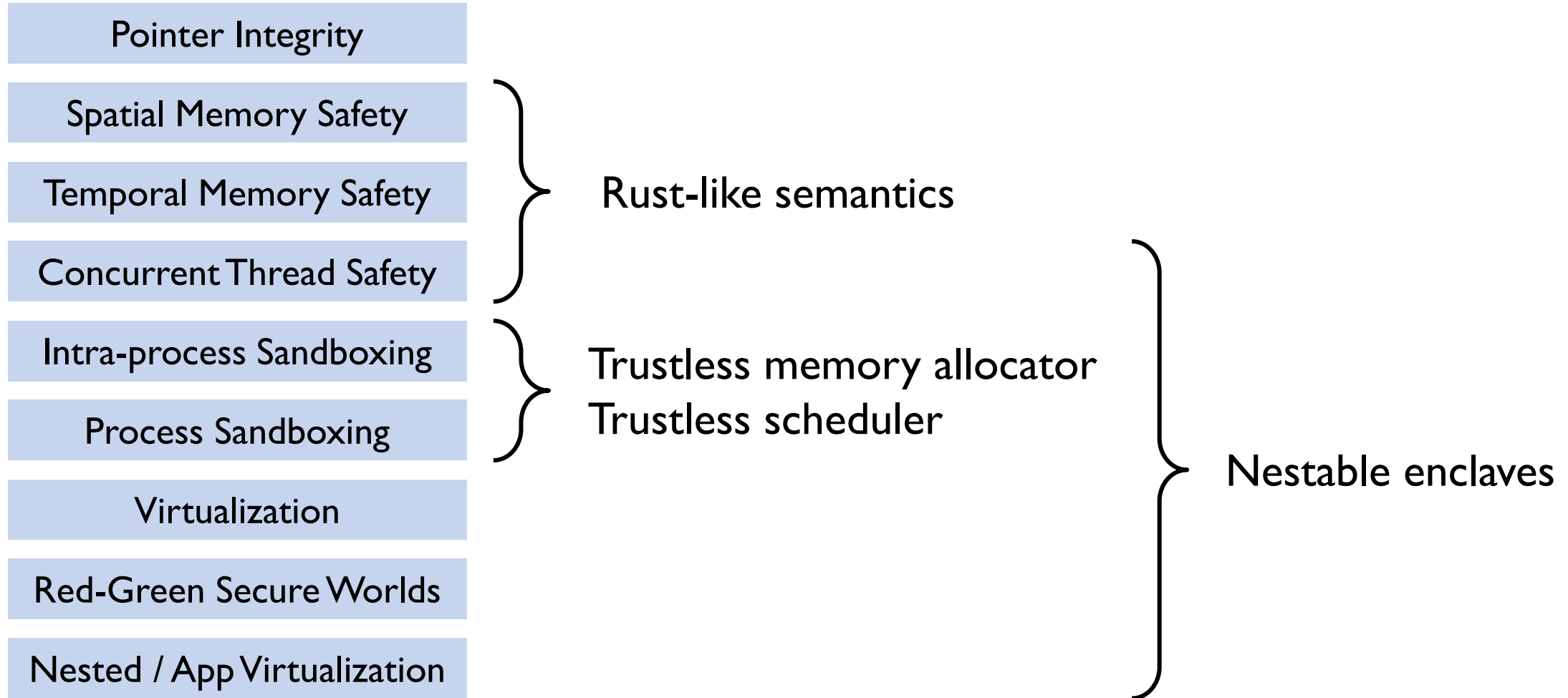




# Case Study: Nestable Enclaves

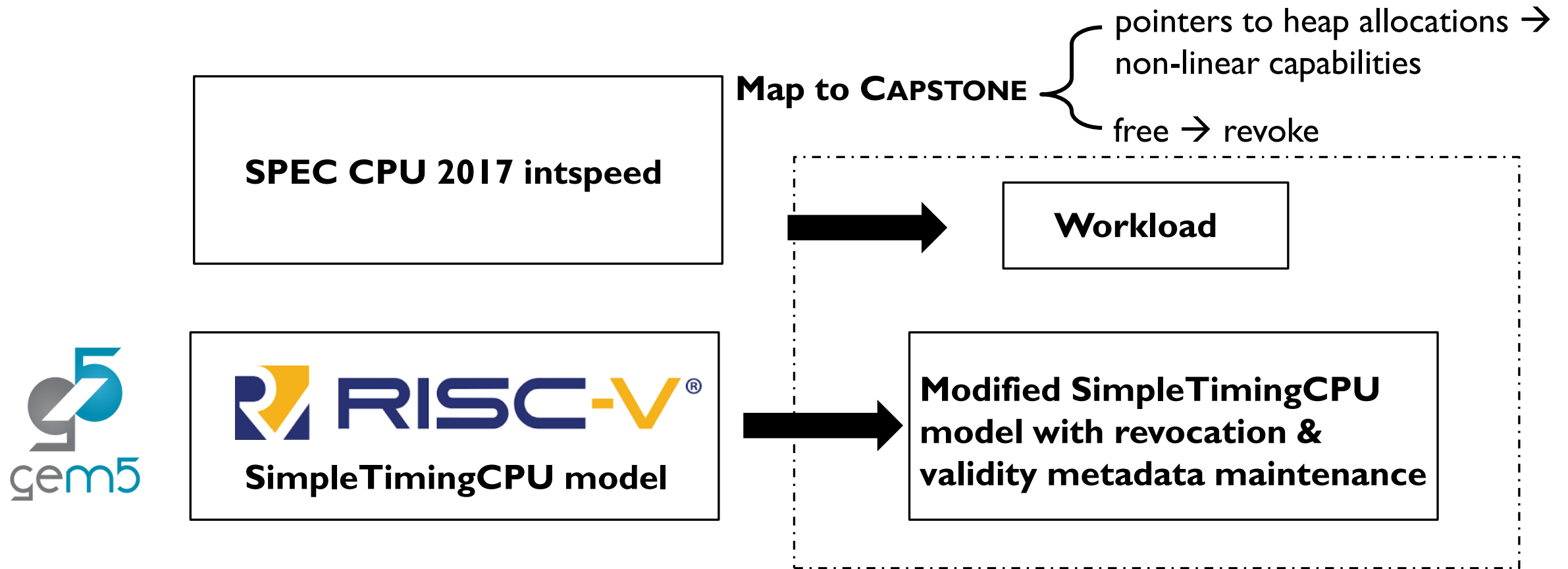


# Case Studies



Takeaway: CAPSTONE is highly expressive

# Preliminary Performance Evaluation



**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!