S C I E N C E ▪ T E C H N O L O G Y ▪ P A S S I O N

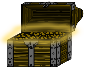
TIMBER-V

Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V

Samuel Weiser[*] Mario Werner[*]
Ferdinand Brasser[†] Maja Malenko[*]
Stefan Mangard[*] Ahmad Sadeghi[†]
[*]Graz University of Technology
[†]TU Darmstadt

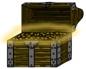
Motivation

- Goal: Protect sensitive code and data



Motivation

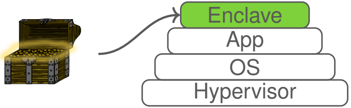
- Goal: Protect sensitive code and data



App
OS
Hypervisor

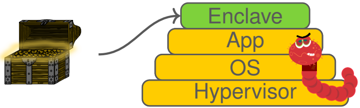
Motivation

- Goal: Protect sensitive code and data



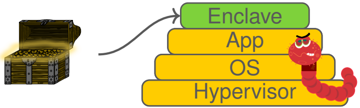
Motivation

- Goal: Protect sensitive code and data from malicious software

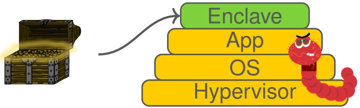


Motivation

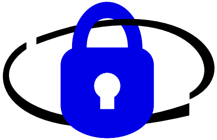
- Goal: Protect sensitive code and data from malicious software
- Intel SGX is cool, but



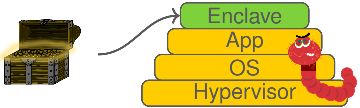
Motivation



- Goal: Protect sensitive code and data from malicious software
- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]



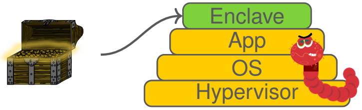
Motivation



- Goal: Protect sensitive code and data from malicious software
- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]
- RISC-V is completely open

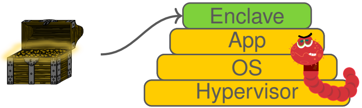
Motivation

- Goal: Protect sensitive code and data from malicious software
- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]
- RISC-V is completely open
  - **Bring SGX features to embedded RISC-V**

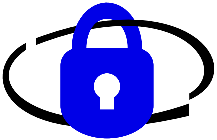


RISC-V®

Motivation



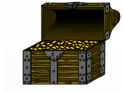
- Goal: Protect sensitive code and data from malicious software
- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]



- RISC-V is completely open
  - **Bring SGX features to embedded RISC-V** (embedded = ARM Cortex-M)

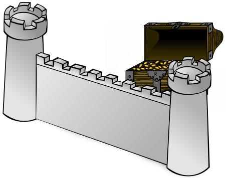
Background: Enclaves

- Secure execution



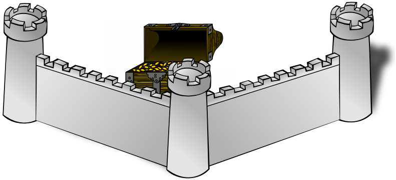
Background: Enclaves

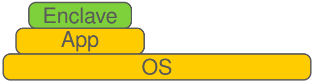
- Secure execution
- Protect against all other software
    - Malicious app



Background: Enclaves

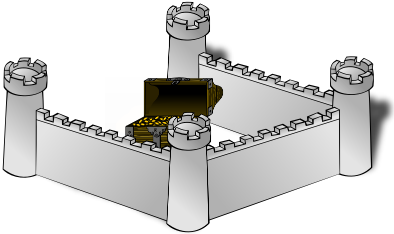
- Secure execution
- Protect against all other software
  - Malicious app
  - Malicious OS

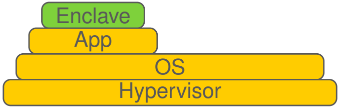




Background: Enclaves

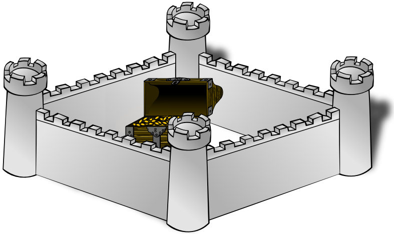
- Secure execution
- Protect against all other software
    - Malicious app
    - Malicious OS
    - Malicious hypervisor

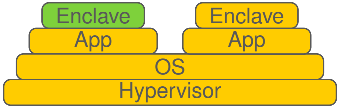




Background: Enclaves

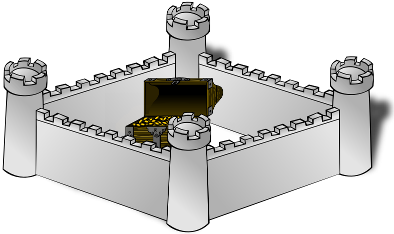
- Secure execution
- Protect against all other software
    - Malicious app
    - Malicious OS
    - Malicious hypervisor
    - Malicious enclaves

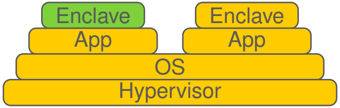




Background: Enclaves

- Secure execution
- Protect against all other software
    - Malicious app
    - Malicious OS
    - Malicious hypervisor
    - Malicious enclaves
- Minimal trust (enclave + HW)





Related Work - Secure Execution

- ”Large”: Sanctum (RISC-V) [2]
- ”Embedded”: [3, 4, 7, 1, 9]
    - RISC-V: MultiZone [5], Keystone [6]

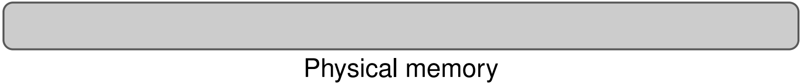
Related Work - Secure Execution

- ”Large”: Sanctum (RISC-V) [2]
- ”Embedded”: [3, 4, 7, 1, 9]
  - RISC-V: MultiZone [5], Keystone [6]

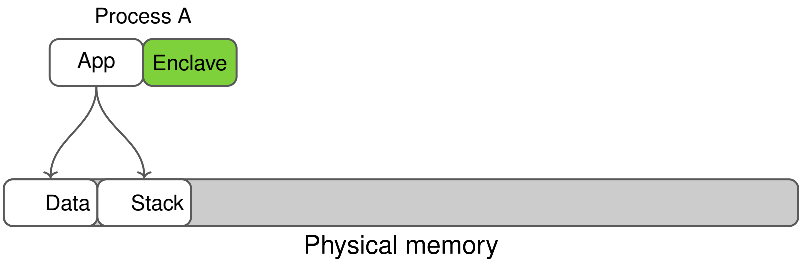
Problems

- Inflexible isolation boundaries
- **Memory fragmentation**

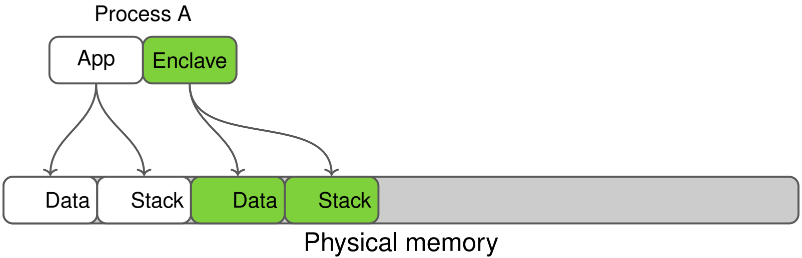
Problem: Memory Fragmentation



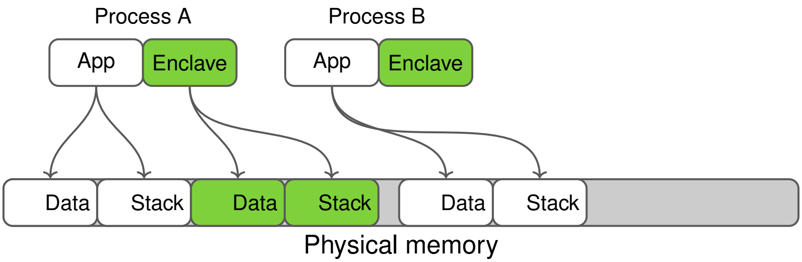
Problem: Memory Fragmentation



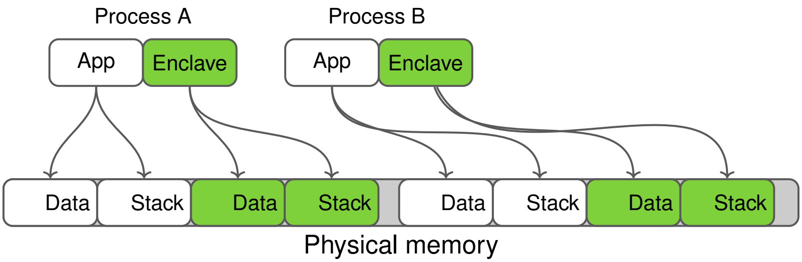
Problem: Memory Fragmentation



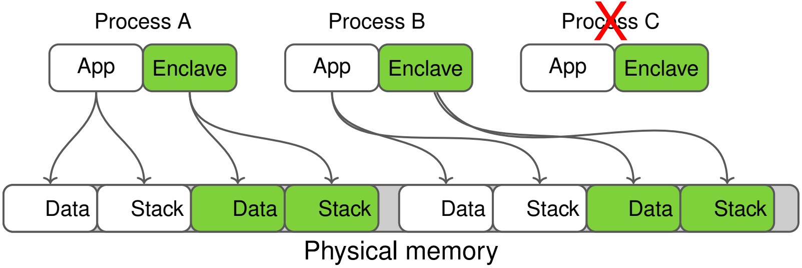
Problem: Memory Fragmentation



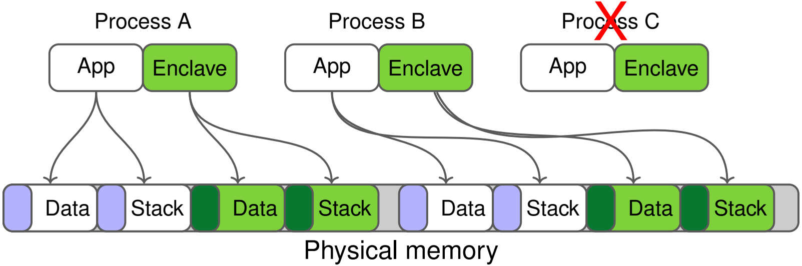
Problem: Memory Fragmentation



Problem: Memory Fragmentation

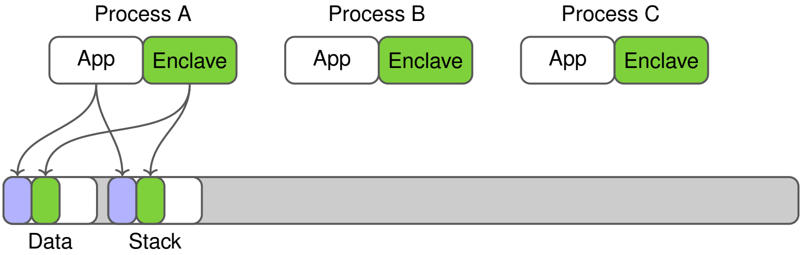


Problem: Memory Fragmentation



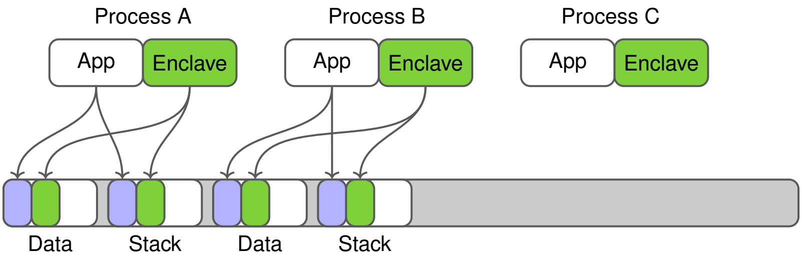
- Bad memory utilization

Problem: Memory Fragmentation



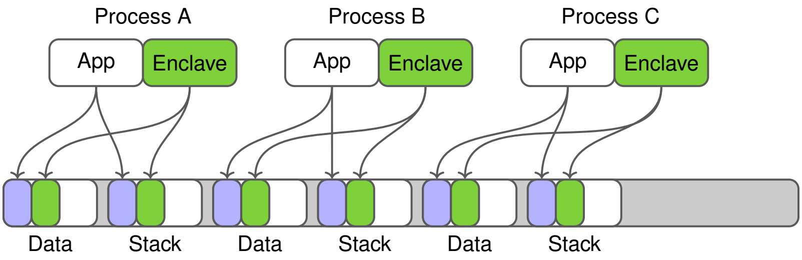
- Bad memory utilization
- We want to interleave memory → Stack sharing

Problem: Memory Fragmentation



- Bad memory utilization
- We want to interleave memory → Stack sharing

Problem: Memory Fragmentation



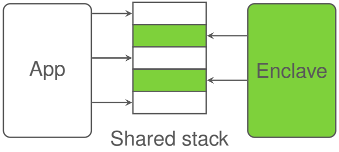
- Bad memory utilization
- We want to interleave memory $\rightarrow$ Stack sharing

Contributions

- TIMBER-V: Enclaves from tagged memory for embedded RISC-V

Contributions

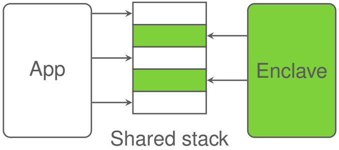
- TIMBER-V: Enclaves from tagged memory for embedded RISC-V
- Novel stack sharing



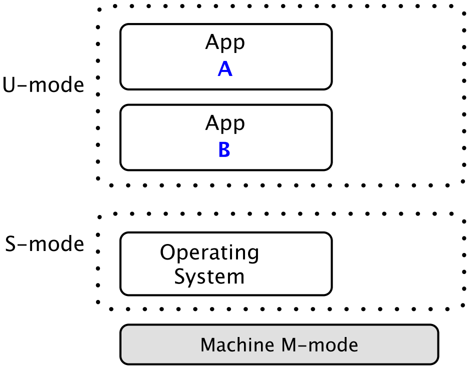
Contributions

- TIMBER-V: Enclaves from tagged memory for embedded RISC-V
- Novel stack sharing
- Fast shared enclave memory

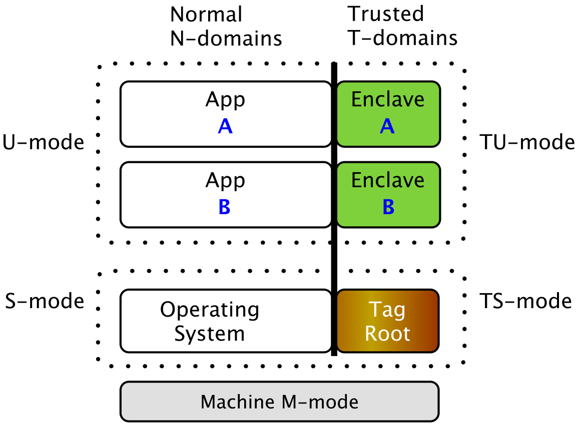
  ...
- Proof-of-concept



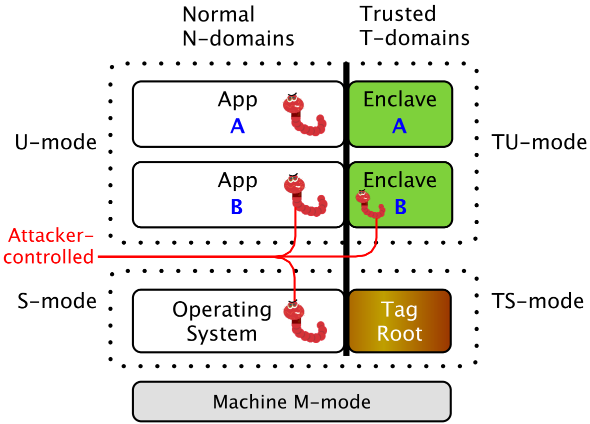
TIMBER-V Overview



TIMBER-V Overview



TIMBER-V Overview



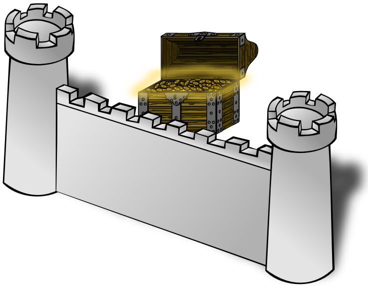
TIMBER-V

Enclave building blocks:

TIMBER-V

Enclave building blocks:

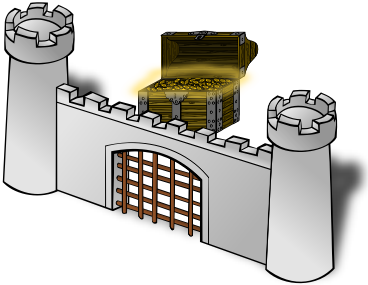
- Memory isolation



TIMBER-V

Enclave building blocks:

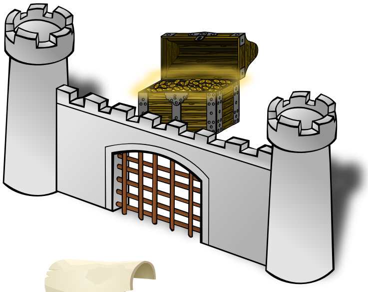
- Memory isolation
- Entry points



TIMBER-V

Enclave building blocks:

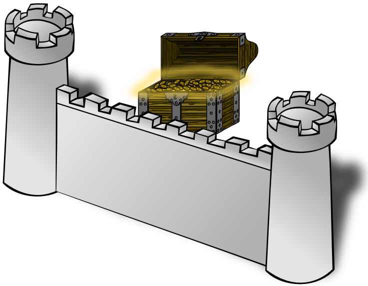
- Memory isolation
- Entry points
- Attestation, sealing
- Inter-enclave communication



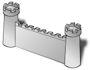
TIMBER-V

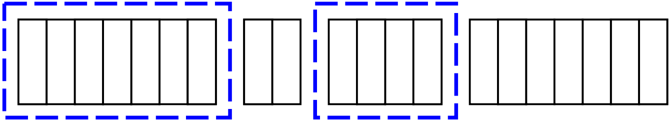
Enclave building blocks:

- **Memory isolation**
- Entry points
- Attestation, sealing
- Inter-enclave communication



Traditional Memory Protection Unit (MPU)

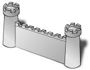




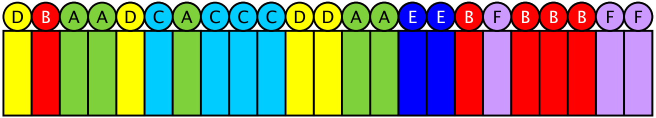
Physical
Memory

- MPU regions define application
- Problem: inflexible

Tagged Memory

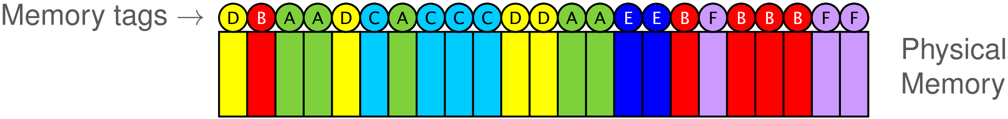


Memory tags →



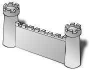
Physical
Memory

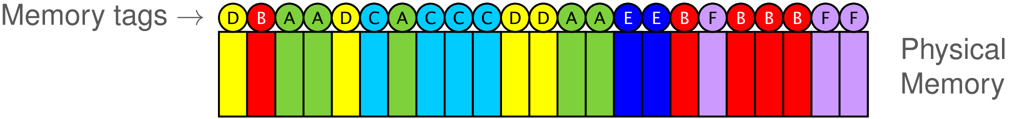
Tagged Memory



- Memory tags define applications

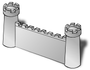
Tagged Memory

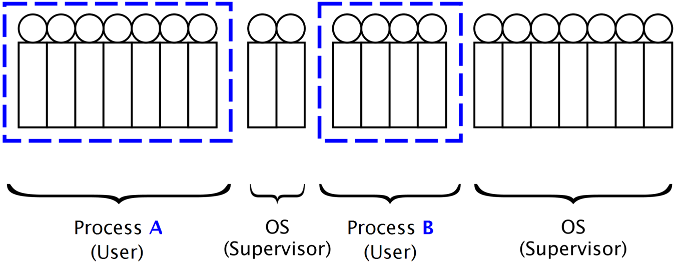




- Memory tags define applications
- Problem: high memory overhead

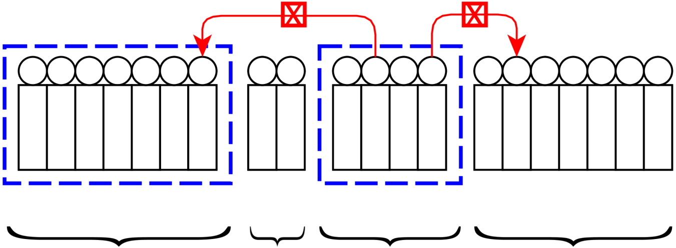
TIMBER-V: MPU + Tagged Memory





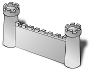
- MPU region defines application

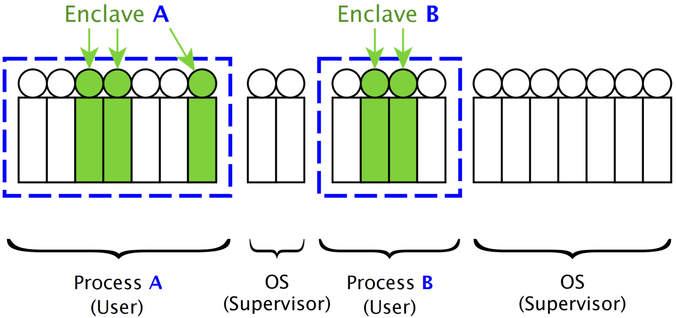
TIMBER-V: MPU + Tagged Memory



- MPU region defines application
- Application cannot escape

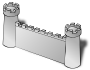
TIMBER-V: MPU + Tagged Memory

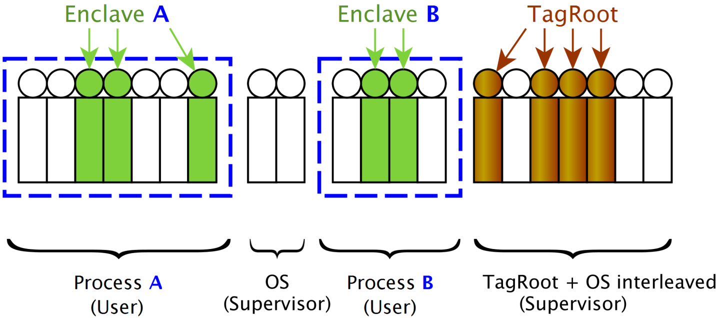




- Memory tag defines enclave

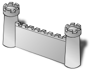
TIMBER-V: MPU + Tagged Memory

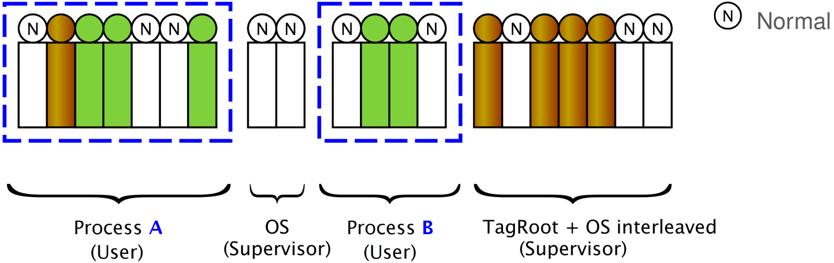




- Memory tag defines enclave
- Memory tag defines TagRoot

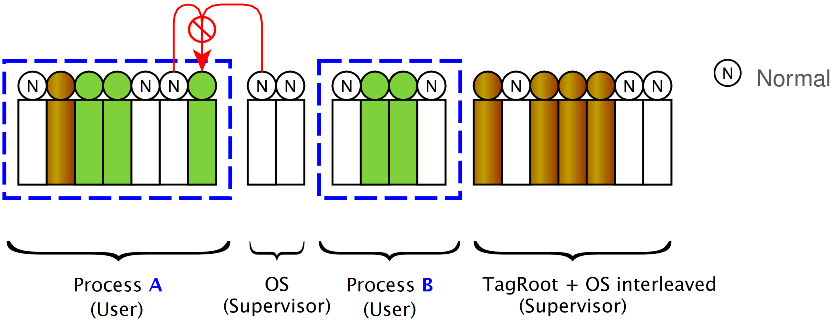
TIMBER-V Tag Isolation





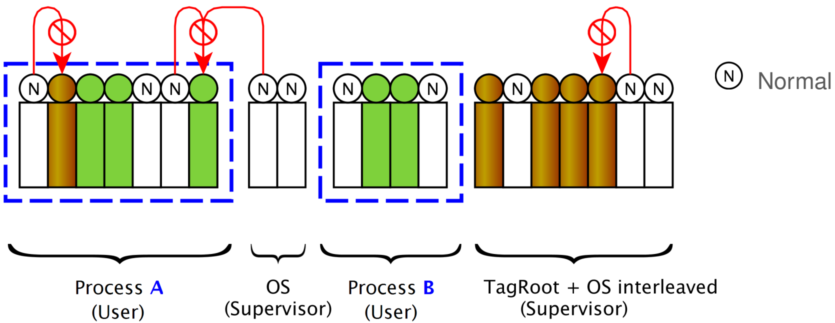
- N Normal memory

TIMBER-V Tag Isolation



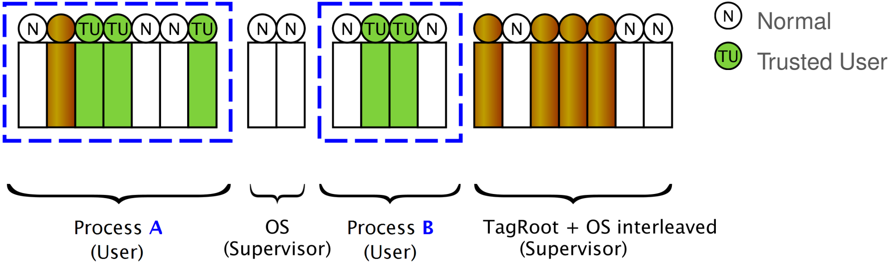
- N Normal memory
- Cannot access others

TIMBER-V Tag Isolation



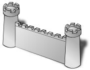
- N Normal memory
- Cannot access others

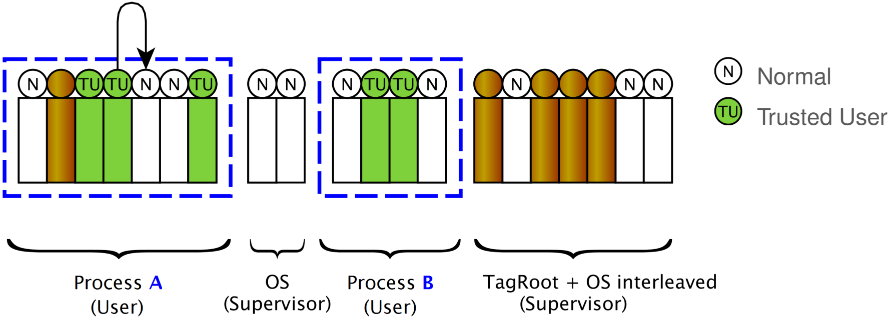
TIMBER-V Tag Isolation



- TU Trusted User memory (enclaves)

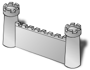
TIMBER-V Tag Isolation

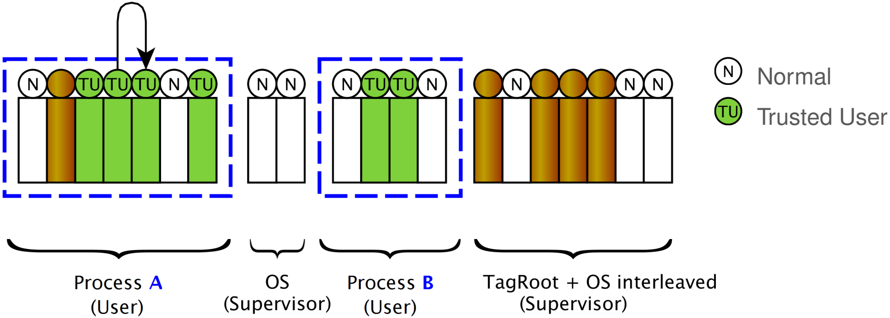




- TU Trusted User memory (enclaves)
- Can access and update normal memory

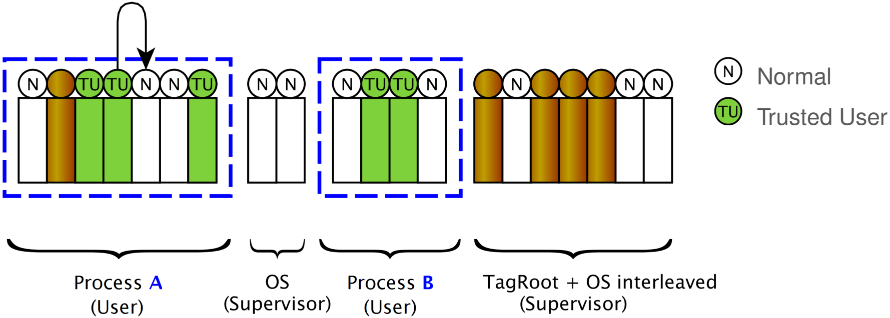
TIMBER-V Tag Isolation





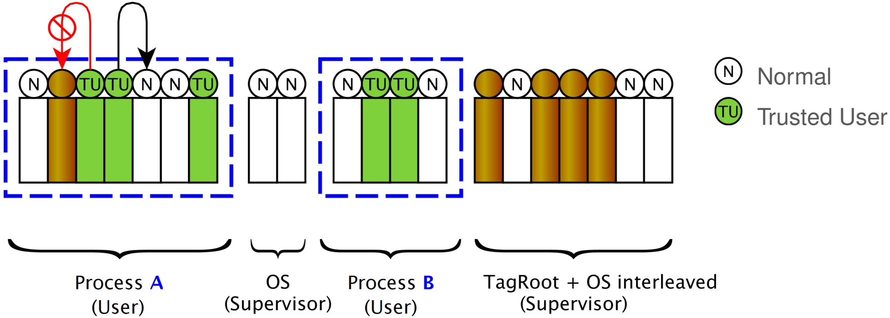
- TU Trusted User memory (enclaves)
- Can access and update normal memory

TIMBER-V Tag Isolation



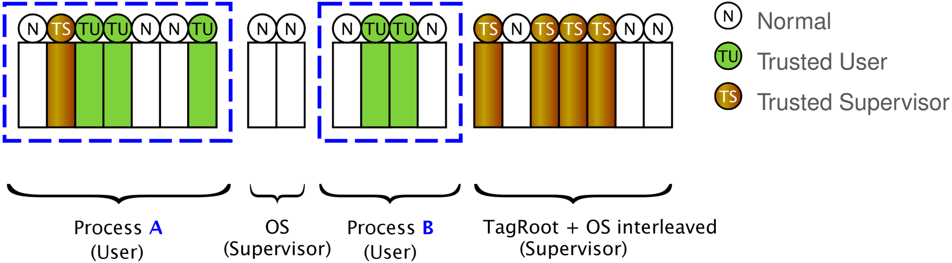
- TU Trusted User memory (enclaves)
- Can access and update normal memory

TIMBER-V Tag Isolation



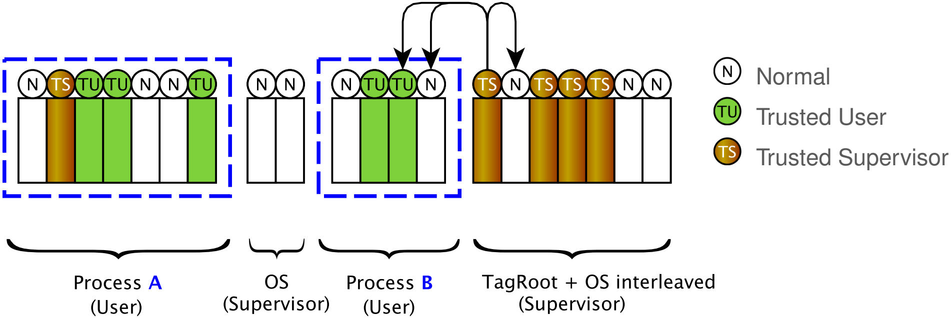
- TU Trusted User memory (enclaves)
- Can access and update normal memory

TIMBER-V Tag Isolation



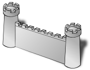
- TS Trusted Supervisor memory (TagRoot)

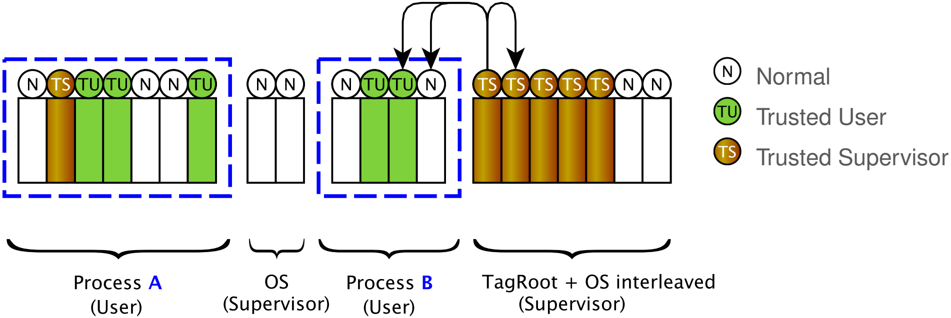
TIMBER-V Tag Isolation



- TS Trusted Supervisor memory (TagRoot)
- Can access and update any tag

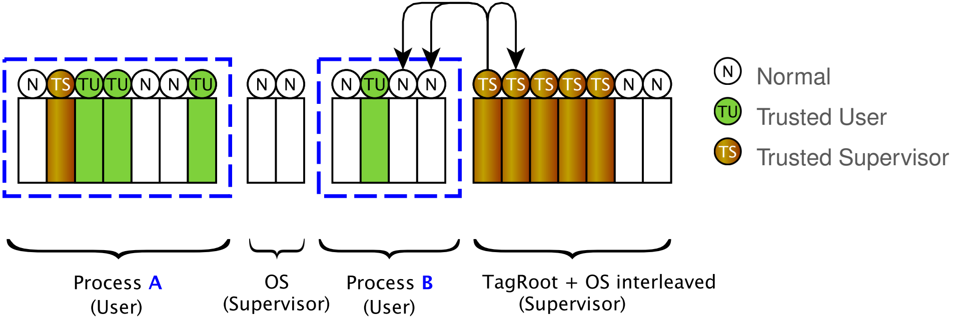
TIMBER-V Tag Isolation





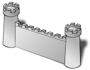
- TS Trusted Supervisor memory (TagRoot)
- Can access and update any tag

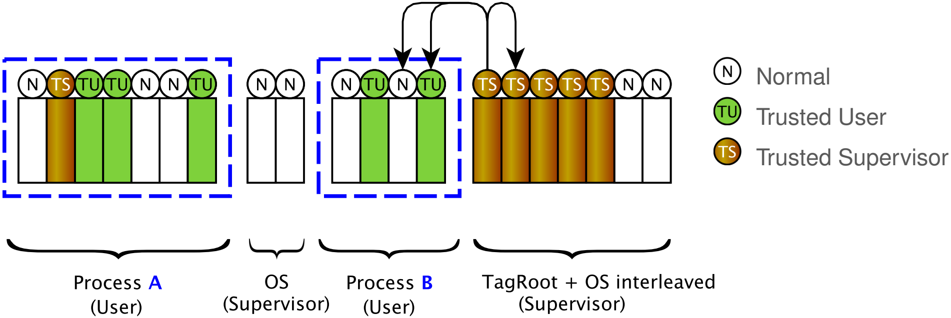
TIMBER-V Tag Isolation



- TS Trusted Supervisor memory (TagRoot)
- Can access and update any tag

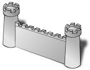
TIMBER-V Tag Isolation

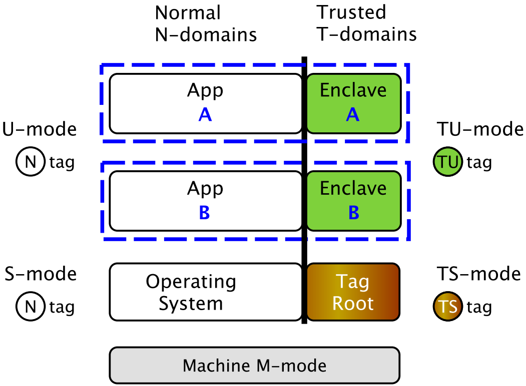




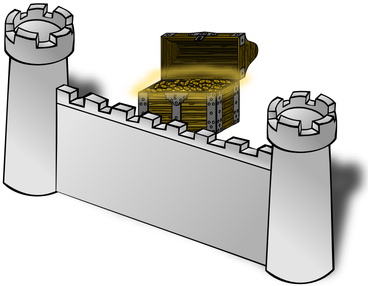
- TS Trusted Supervisor memory (TagRoot)
- Can access and update any tag

TIMBER-V Memory Isolation





TIMBER-V



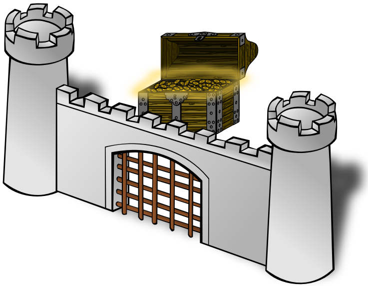
Enclave building blocks:

- ✓ Memory isolation
- Entry points
- Attestation, sealing
- Inter-enclave communication

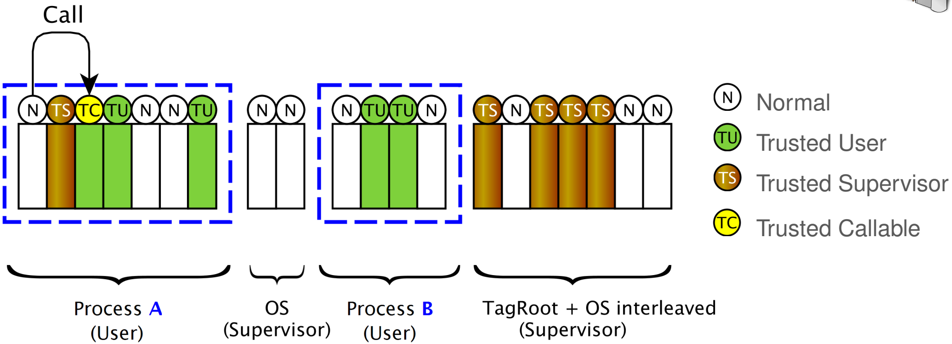
TIMBER-V

Enclave building blocks:

- ✓ Memory isolation
- **Entry points**
- Attestation, sealing
- Inter-enclave communication

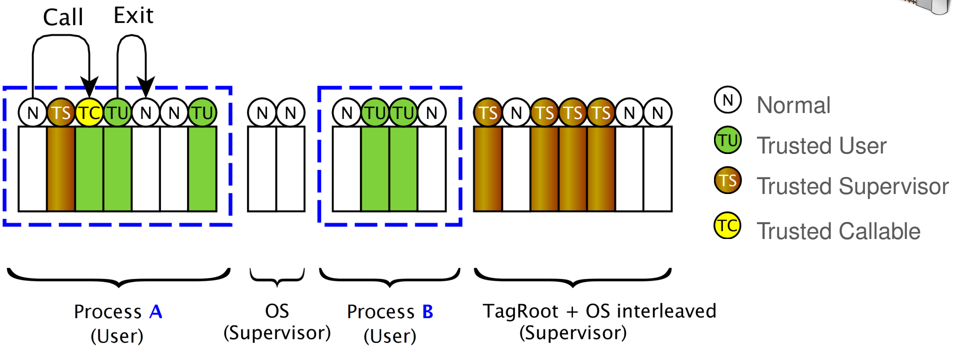


TIMBER-V Entry Points



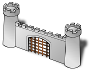
- Enter only at TC Trusted Callable

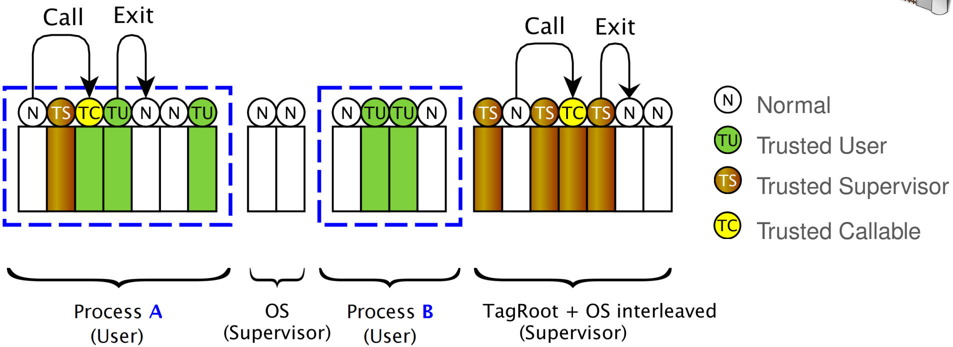
TIMBER-V Entry Points



- Enter only at TC Trusted Callable
- Zero runtime overhead (ordinary `jmp`)

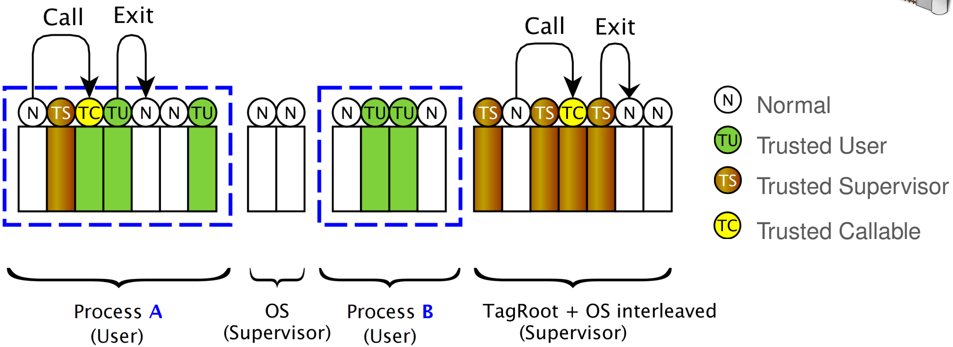
TIMBER-V Entry Points





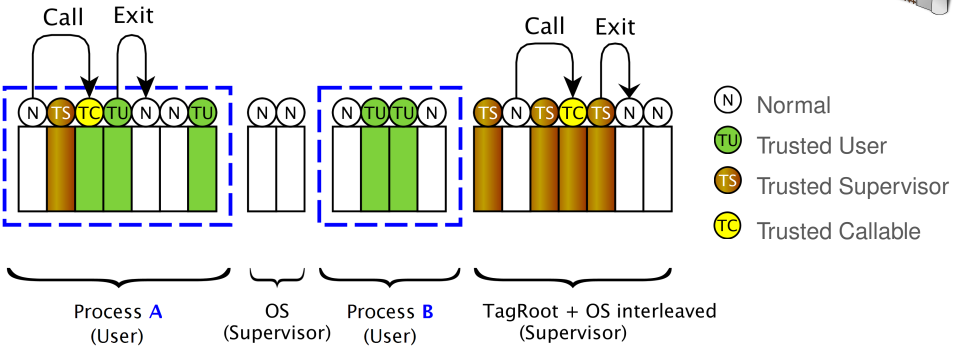
- Enter only at TC Trusted Callable
- Zero runtime overhead (ordinary `jmp`)

TIMBER-V Entry Points



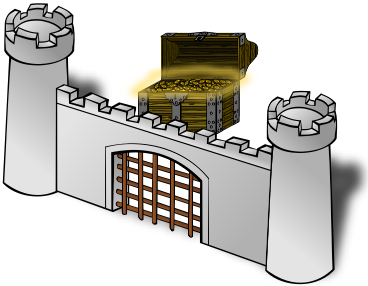
- Four tags $\rightarrow$ two tag bits only

TIMBER-V Entry Points



- Four tags $\rightarrow$ two tag bits only
- For 32-bit system +6.25% memory overhead

TIMBER-V



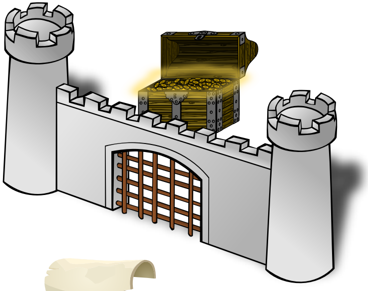
Enclave building blocks:

- ✓ Memory isolation
- ✓ Entry points
- Attestation, sealing
- Inter-enclave communication

TIMBER-V

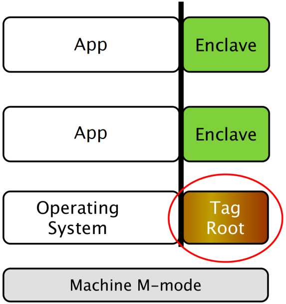
Enclave building blocks:

- ✓ Memory isolation
- ✓ Entry points
- **Attestation, sealing**
- **Inter-enclave communication**



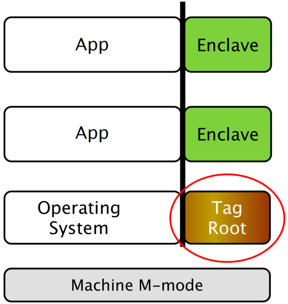
TIMBER-V TagRoot

- Root of trust in privileged software



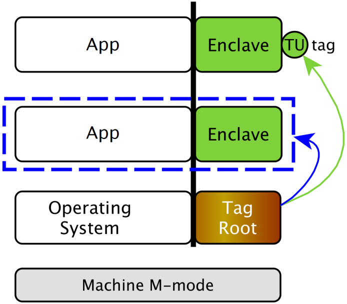
TIMBER-V TagRoot

- Root of trust in privileged software
- Supports SGX and TrustZone model



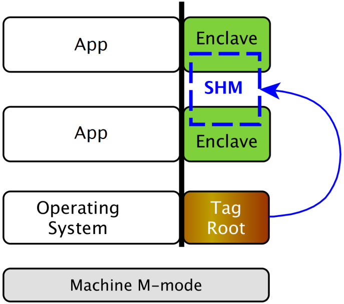
TIMBER-V TagRoot

- Root of trust in privileged software
- Supports SGX and TrustZone model
- Enclave management



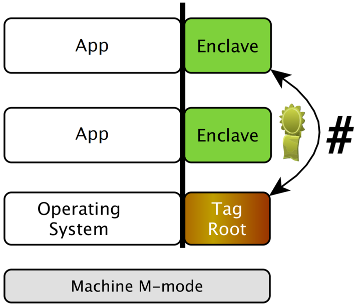
TIMBER-V TagRoot

- Root of trust in privileged software
- Supports SGX and TrustZone model
- Enclave management
- Inter-enclave communication
  - Fast shared memory
  - Mutual authentication
  - Implicit local attestation



TIMBER-V TagRoot

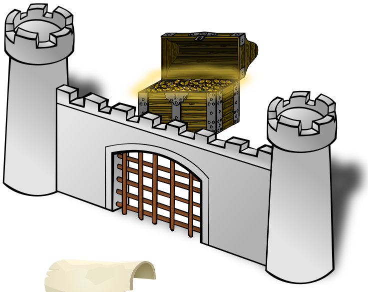
- Root of trust in privileged software
- Supports SGX and TrustZone model
- Enclave management
- Inter-enclave communication
    - Fast shared memory
    - Mutual authentication
    - Implicit local attestation
- Sealing (like SGX)



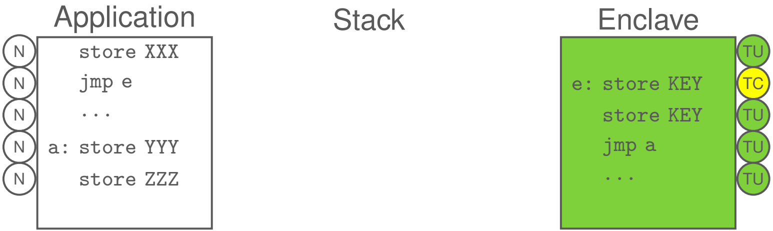
TIMBER-V

Enclave building blocks:

- ✓ Memory isolation
- ✓ Entry points
- ✓ Attestation, sealing
- ✓ Inter-enclave communication

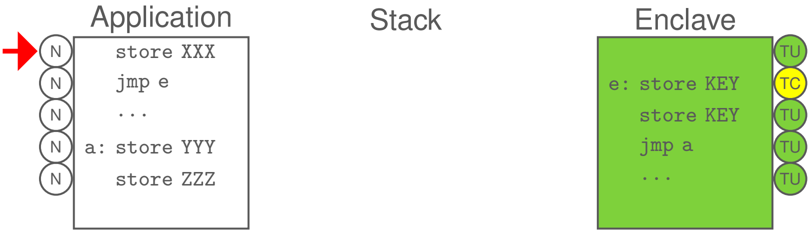


Novel Stack Sharing



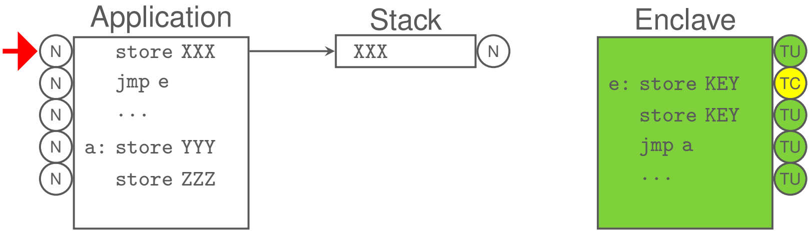
- Single stack shared between application and enclave ...

Novel Stack Sharing



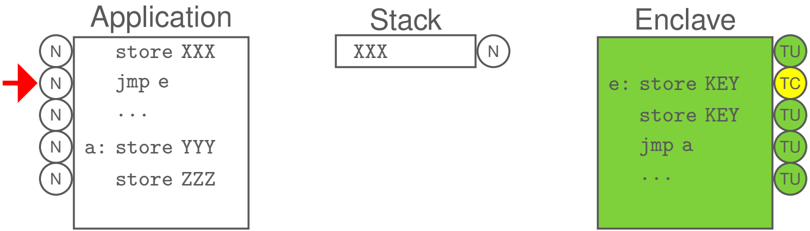
- Single stack shared between application and enclave ...

Novel Stack Sharing



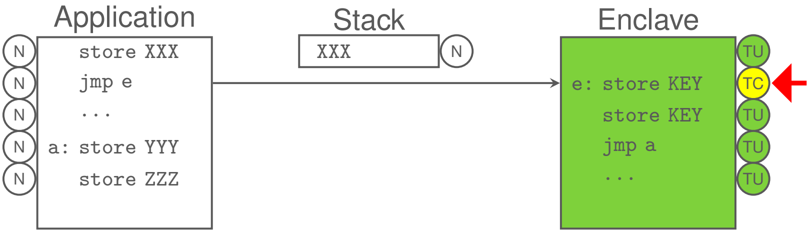
- Single stack shared between application and enclave ...

Novel Stack Sharing



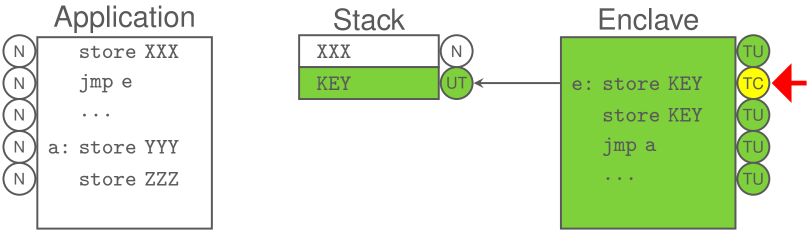
- Single stack shared between application and enclave ...

Novel Stack Sharing



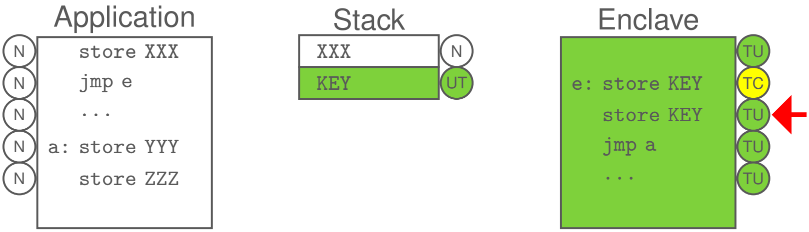
- Single stack shared between application and enclave ...

Novel Stack Sharing



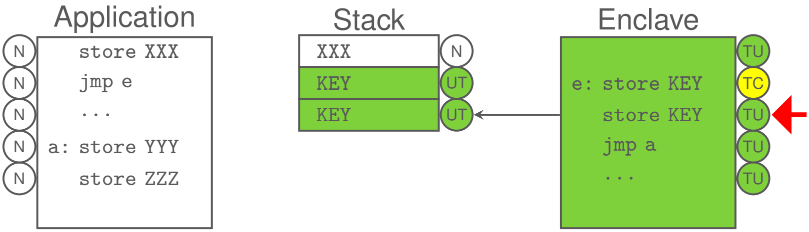
- Single stack shared between application and enclave ...

Novel Stack Sharing



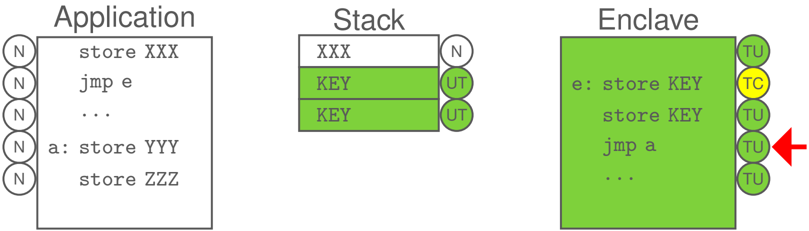
- Single stack shared between application and enclave ...

Novel Stack Sharing



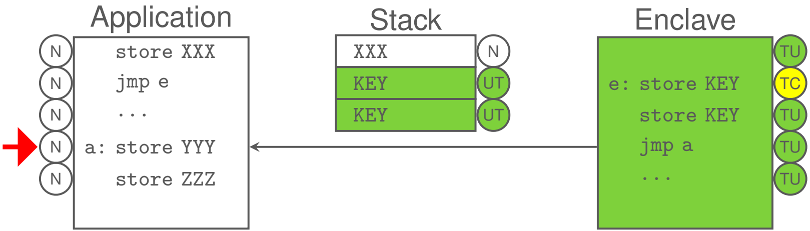
- Single stack shared between application and enclave ...

Novel Stack Sharing



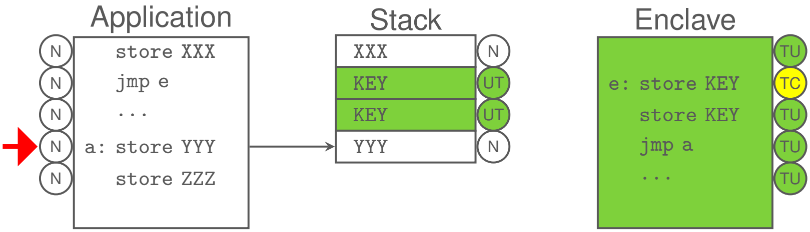
- Single stack shared between application and enclave ...

Novel Stack Sharing



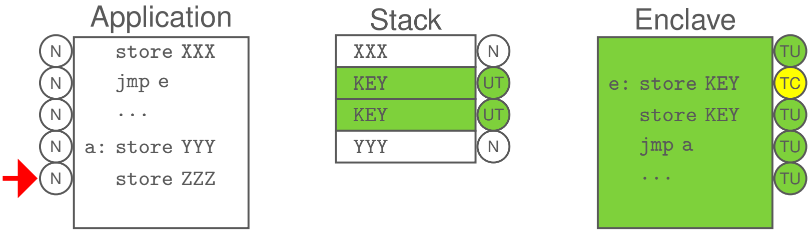
- Single stack shared between application and enclave ...

Novel Stack Sharing



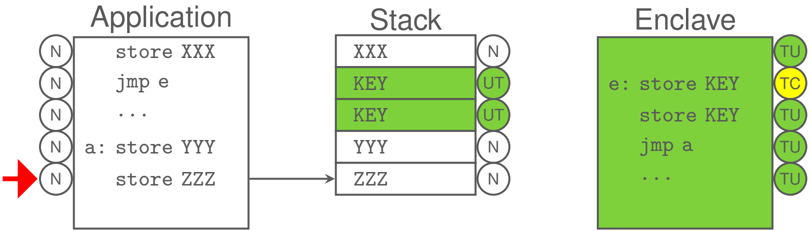
- Single stack shared between application and enclave ...

Novel Stack Sharing



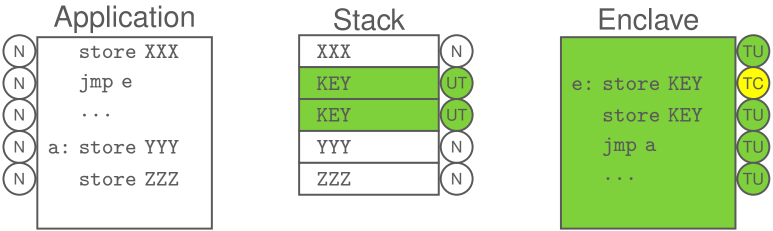
- Single stack shared between application and enclave ...

Novel Stack Sharing



- Single stack shared between application and enclave ...

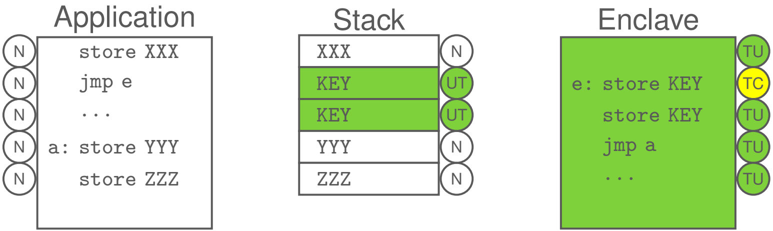
Novel Stack Sharing



- Single stack shared between application and enclave ...

  ... and between TagRoot!

Novel Stack Sharing



- Single stack shared between application and enclave ...
  ... and between TagRoot!
- Heap sharing quite similar

Key Insights

- Build enclaves with tagged memory
  - Fine granularity and high flexibility
- Combination with MPU allows tiny 2-bit tags
- Reduced memory fragmentation
  - Shared stacks, heaps ...

!

Proof-of-Concept

- Integration in ISA simulator (Spike)
- Full TagRoot implementation
- FreeRTOS integration
- Gnu GCC support
- Benchmarks (Coremark, Beebs)
- Open source: `github.com/IAIK/timber-v`

TU Graz

TIMBER-V

Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V

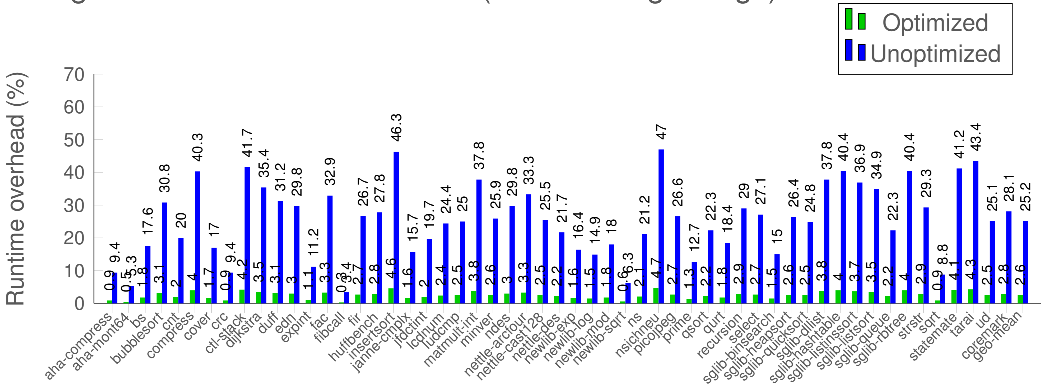
Samuel Weiser[*] Mario Werner[*]
Ferdinand Brasser[†] Maja Malenko[*]
Stefan Mangard[*] Ahmad Sadeghi[†]
[*]Graz University of Technology
[†]TU Darmstadt

Bonus: New Instructions

	RISC-V	TIMBER-V	Arguments
Load	lb,lbu	lbct,lbuct	etag ← check for expected memory tag
	lh,lhu	lhct,lhuct	etag (fault on mismatch)
	lw	lwct	etag
		ltt	etag ← load and test tag w.o. fault
Store	sb	sbct	etag,ntag ← also store new memory tag
	sh	shct	etag,ntag
	sw	swct	etag,ntag

Bonus: TIMBER-V Overhead Estimate

Average overhead: 25.2% vs 2.6% (with caching of tags)



References

[1] Franz Ferdinand Brasser, Brahim El Mahjoub, Ahmad-Reza Sadeghi, Christian Wachsmann, and Patrick Koeberl. "TyTAN: tiny trust anchor for tiny devices". In: Design Automation Conference – DAC'15. ACM, 2015, 34:1–34:6. ISBN: 978-1-4503-3520-1.

[2] Victor Costan, Ilia A. Lebedev, and Srinivas Devadas. "Sanctum: Minimal Hardware Extensions for Strong Software Isolation". In: USENIX Security'16. USENIX Association, 2016, pp. 857–874.

[3] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon, and Daniele Perito. "SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust". In: Network and Distributed System Security Symposium – NDSS'12. The Internet Society, 2012.

[4] Johannes Götzfried, Tilo Müller, Ruan de Clercq, Pieter Maene, Felix C. Freiling, and Ingrid Verbauwhede. "Soteria: Offline Software Protection within Low-cost Embedded Devices". In: Annual Computer Security Applications Conference – ACSAC'15. ACM, 2015, pp. 241–250. ISBN: 978-1-4503-3682-6.

[5] Hex-Five MultiZone Security - the First Trusted Execution Environment (TEE) For RISC-V. https://hex-five.com/products/ (Accessed 2018/12/10). 2018.

[6] Keystone: Open-source Secure Hardware Enclave. https://keystone-enclave.org/ (Accessed 2018/12/10). 2018.

[7] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. "TrustLite: a security architecture for tiny embedded devices". In: European Conference on Computer Systems – EUROSYS'14. ACM, 2014, 10:1–10:14. ISBN: 978-1-4503-2704-6.

[8] Joanna Rutkowska. Thoughts on Intel's upcoming Software Guard Extensions (Part 2). http://theinvisiblethings.blogspot.co.at/2013/09/thoughts-on-intels-upcoming-software.html. (Accessed 2016/10/20). Sept. 2013.

[9] TrustZone Technology for ARMv8-M Architecture. Ref. no. 100690_0200_00_en. https://static.docs.arm.com/100690/0200/armv8m_trustzone_technology_100690_0200.pdf. (Accessed 2018/11/22). 2017.