CURE: A Security Architecture with Customizable and Resilient Enclaves

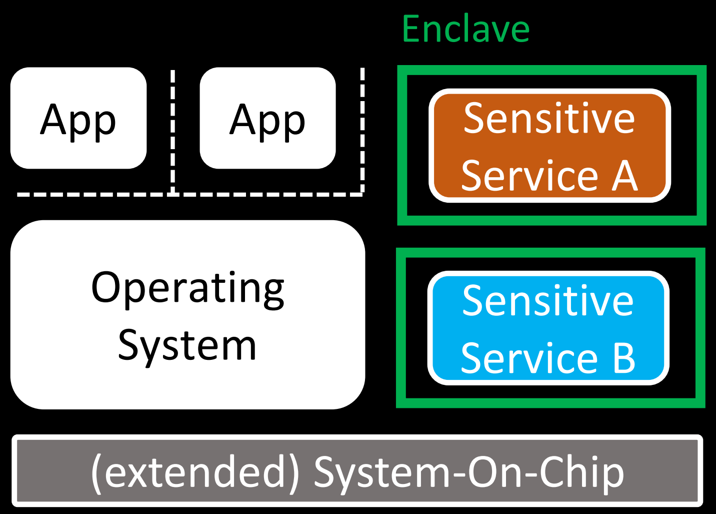
Raad Bahmani, Ferdinand Brasser, Ghada Dessouky, Patrick Jauernig, Matthias Klimmek, Ahmad-Reza Sadeghi, Emmanuel Stapf

Technical University of Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

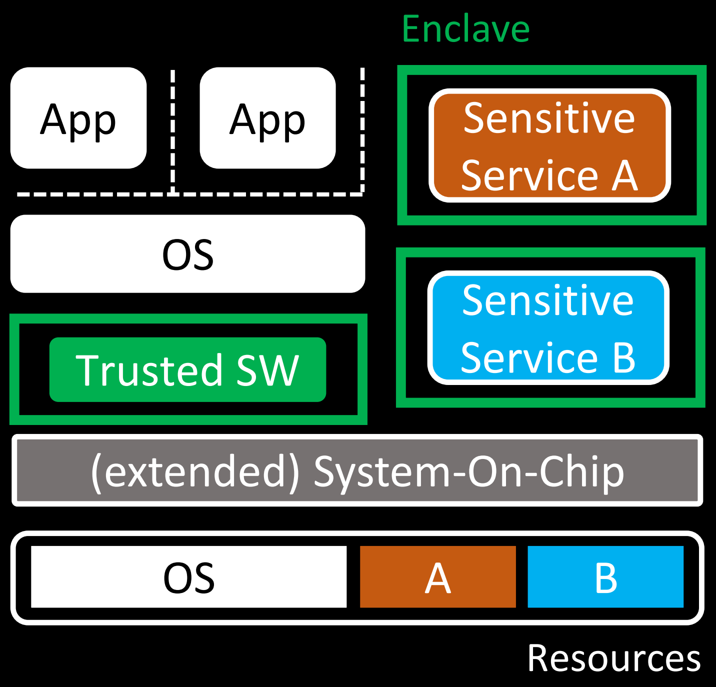
Enclave Security Architectures

- Enclaves prominent approach for protecting sensitive services
- OS assumed to be potentially compromised
- Isolated execution environment, backed by hardware-assisted security mechanisms



Enclave Security Architectures

- Enclaves prominent approach for protecting sensitive services
- OS assumed to be potentially compromised
- Isolated execution environment, backed by hardware-assisted security mechanisms
- Trusted SW configures security mechanisms
- Trusted SW assigns system resources to enclaves (memory, cores, caches)

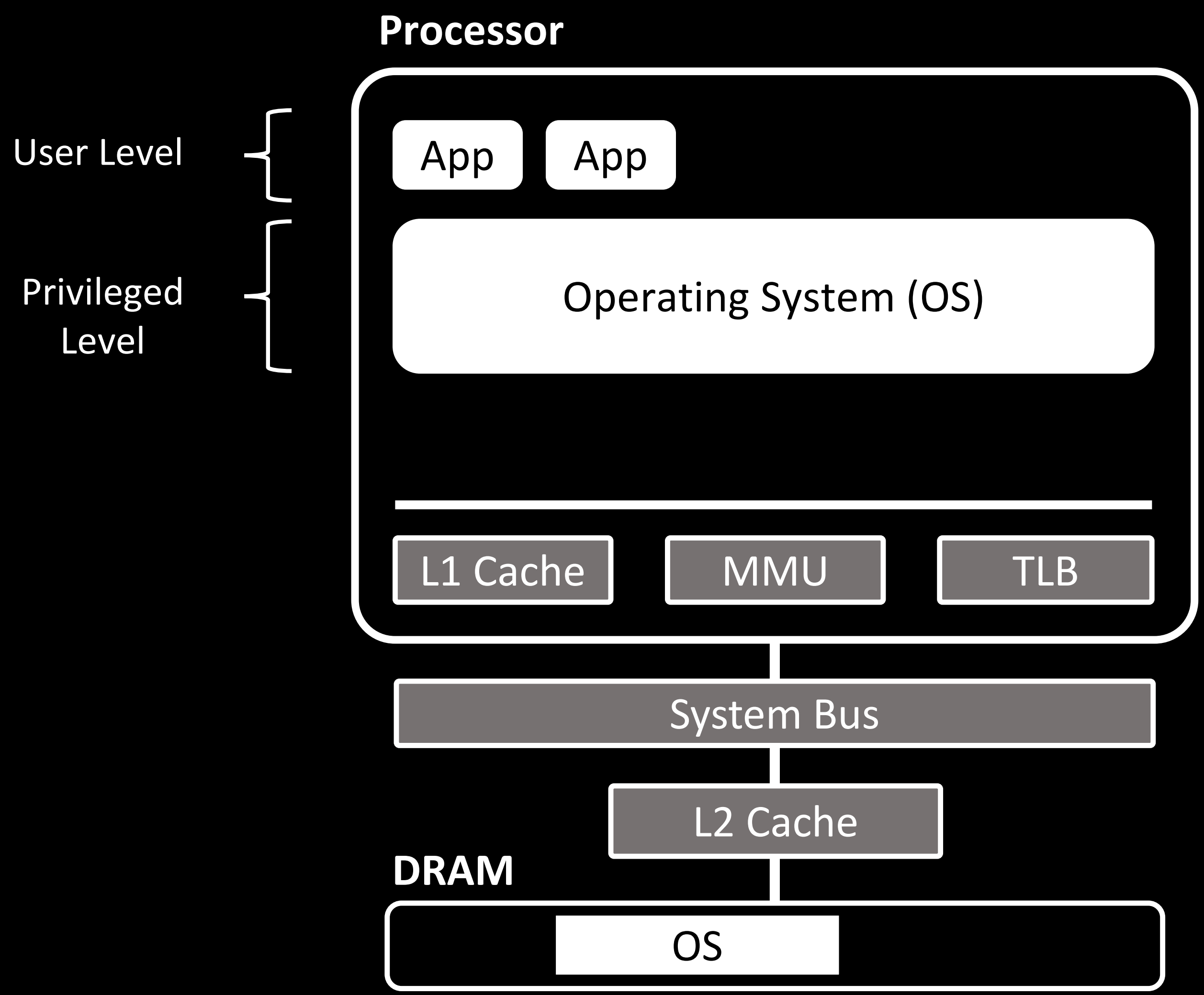


Challenges of Enclave Computing

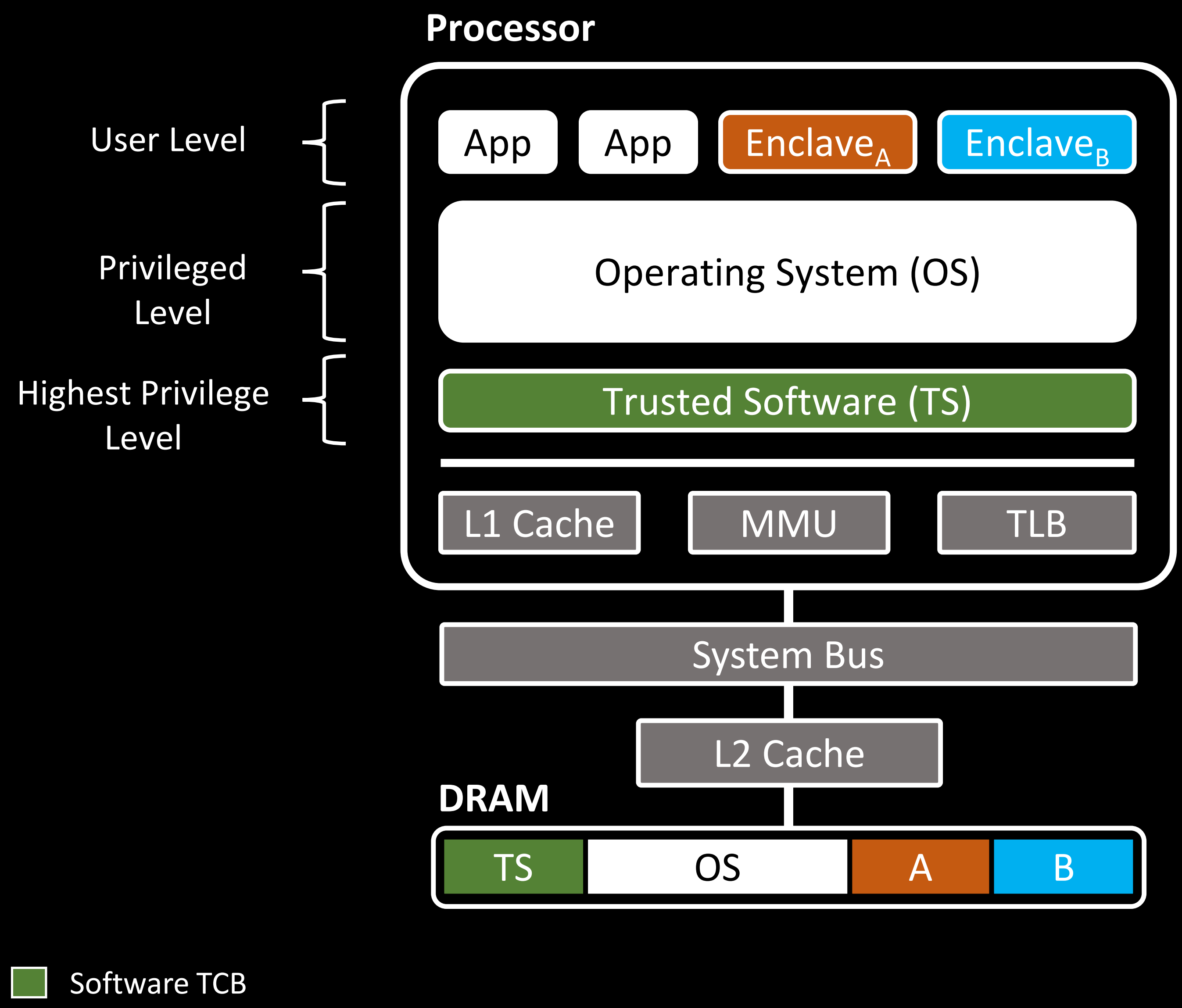
- Security:
  - Side-channels attacks not considered in industry solutions
  - Cache side channels and controlled side channels (page table, interrupt handlers)
- Functionality:
  - Missing functionality regarding secure I/O, secure Direct Memory Access (DMA)
  - Secure binding of enclaves to peripherals
- Configurability:
  - Enclaves cannot be adapted to security and functionality requirements
  - Existing proposals follow *one-size-fits-all* approach

Enclave Types
in Existing Enclave Security Architectures

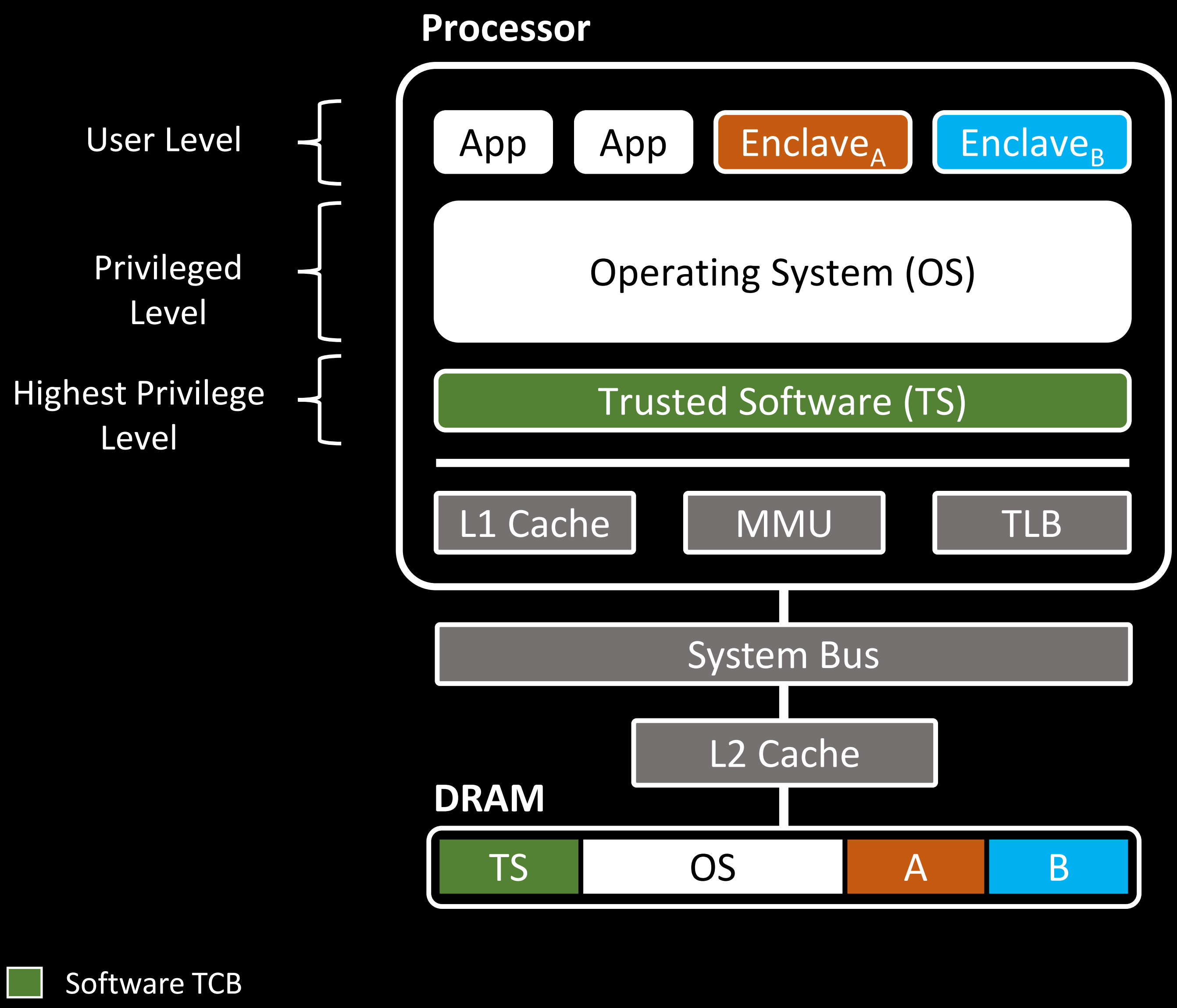
User-space Enclaves



User-space Enclaves



User-space Enclaves



Pros:

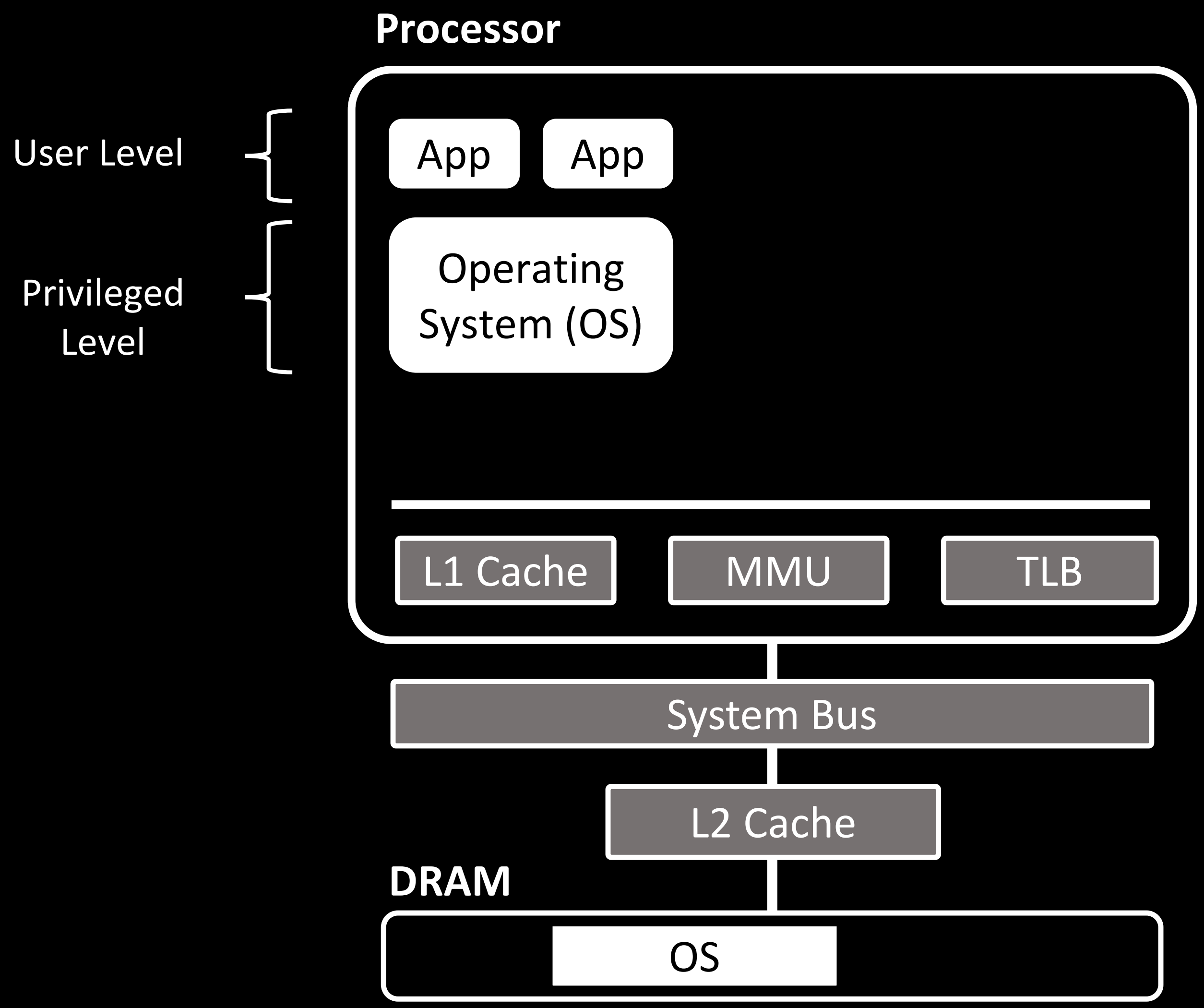
- Reuse of OS functionalities
- Low system resource consumption
- *Easy* to develop

Cons:

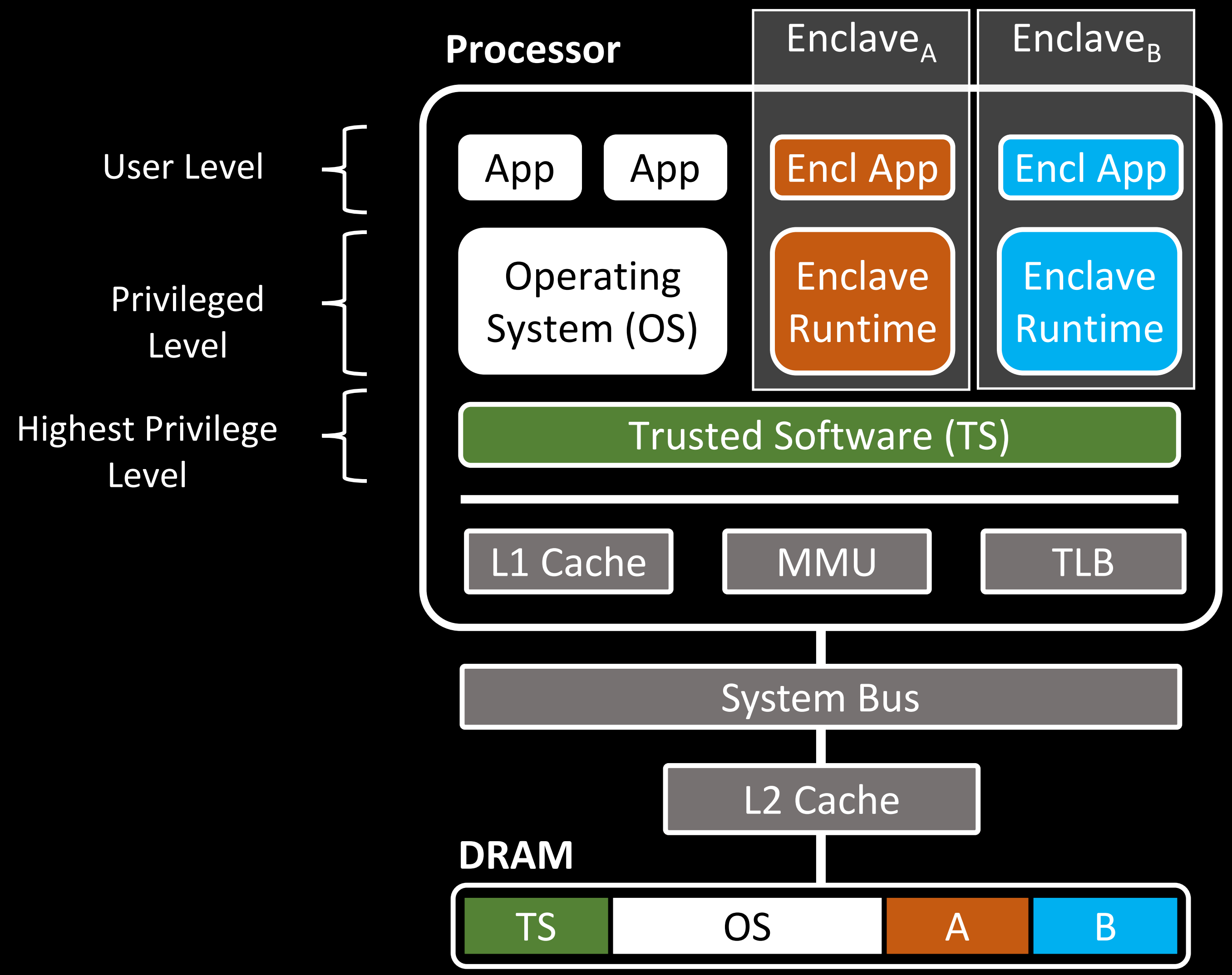
- No privileged code in enclave (I/O)
- Increased performance overhead for context switches
- Protection from controlled side-channel attacks challenging

• Provided by Sanctum [1], SGX [2] and various extensions [3-8]

Kernel-space Enclave

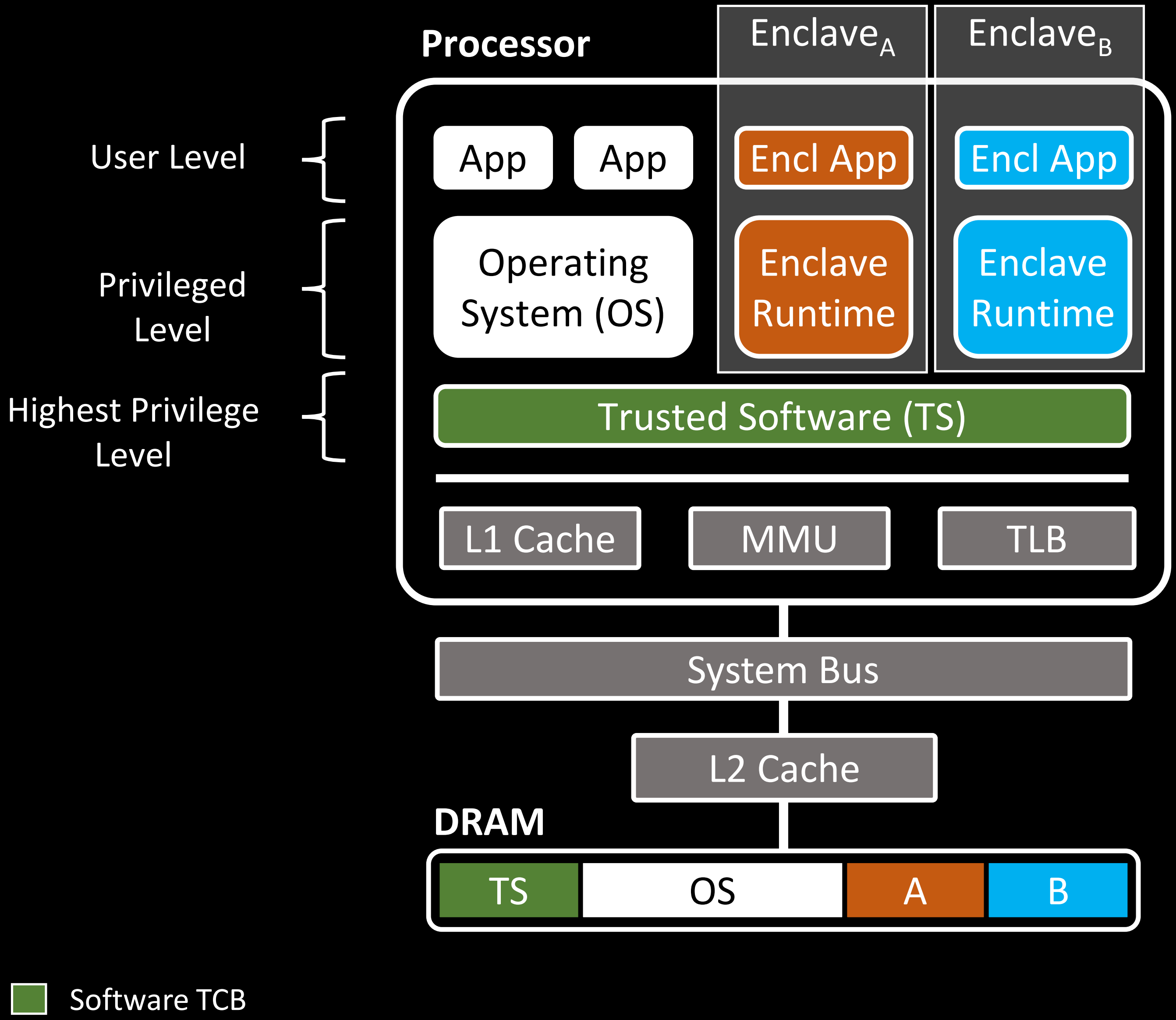


Kernel-space Enclave



Software TCB

Kernel-space Enclave



Pros:

- Privilege code in enclave
- No overhead on context switches
- Easier to prevent controlled side-channel attacks

Cons:

- Increased resource consumption
- Increased overhead for enclave setup
- Need to develop runtime

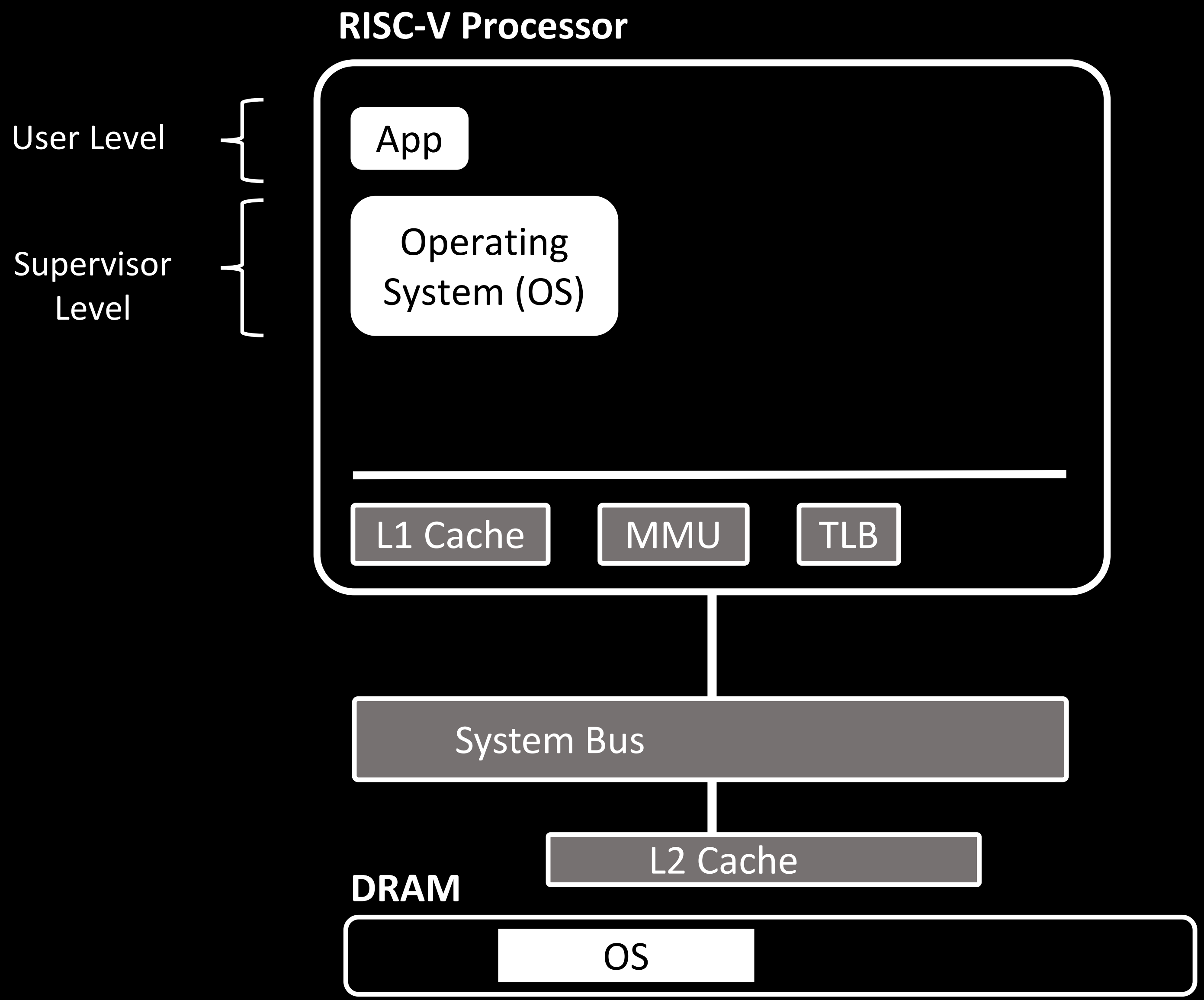
- Provided by TrustZone [9], Sanctuary [10] or Keystone [11]

CURE: A Security Architecture with Customizable and Resilient Enclaves

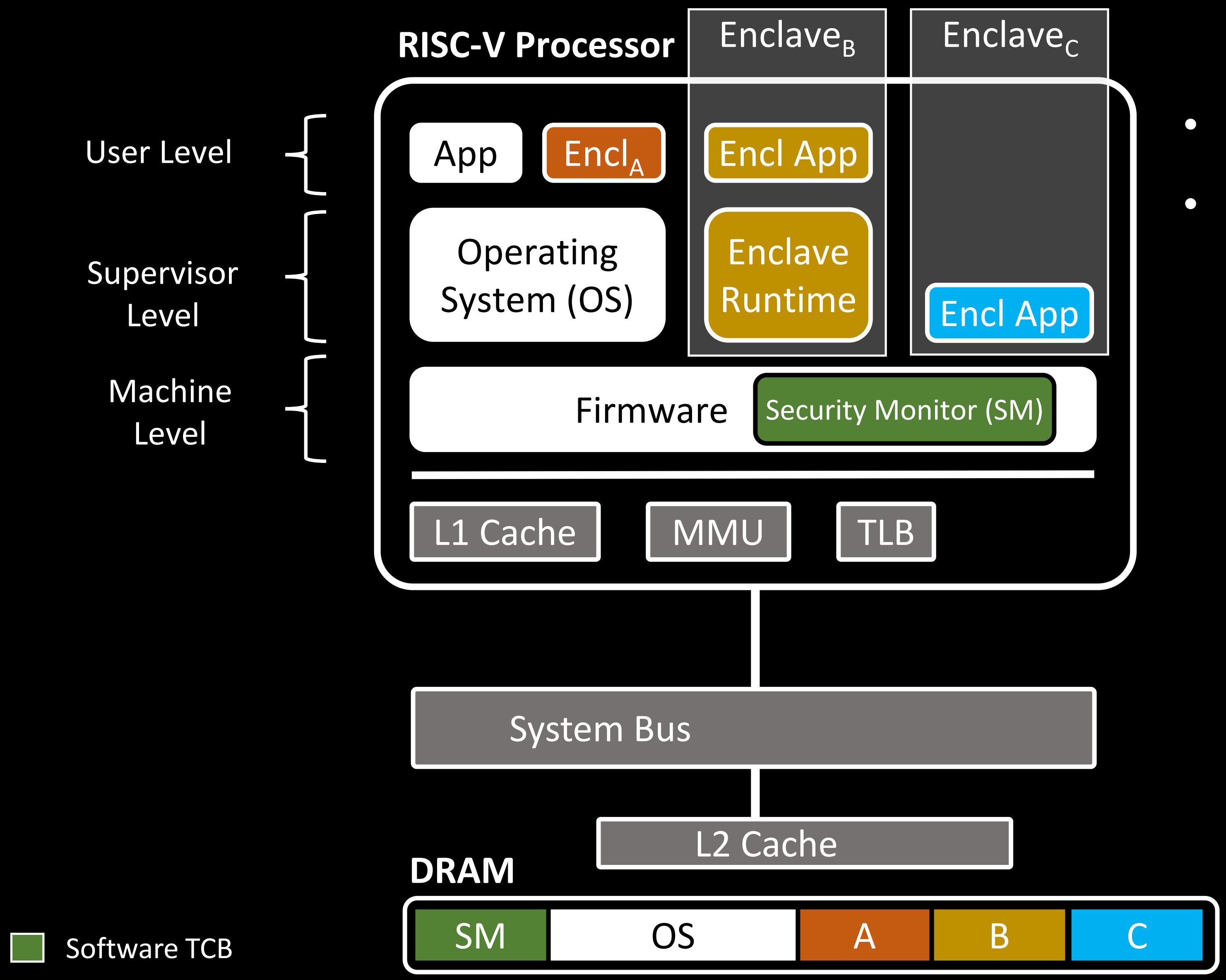
Goals of the CURE Security Architecture

- Tackle the aforementioned challenges
- Security:
  - Protect against controlled side-channel attacks (page table, interrupt handlers)
  - Protect against cache side-channels attacks
- Functionality:
  - Provide a secure binding from peripherals to enclaves
- Configurability:
  - Provides different types of enclave, selected depending on sensitive service & usage scenario

CURE: Customizable and Resilient Enclaves

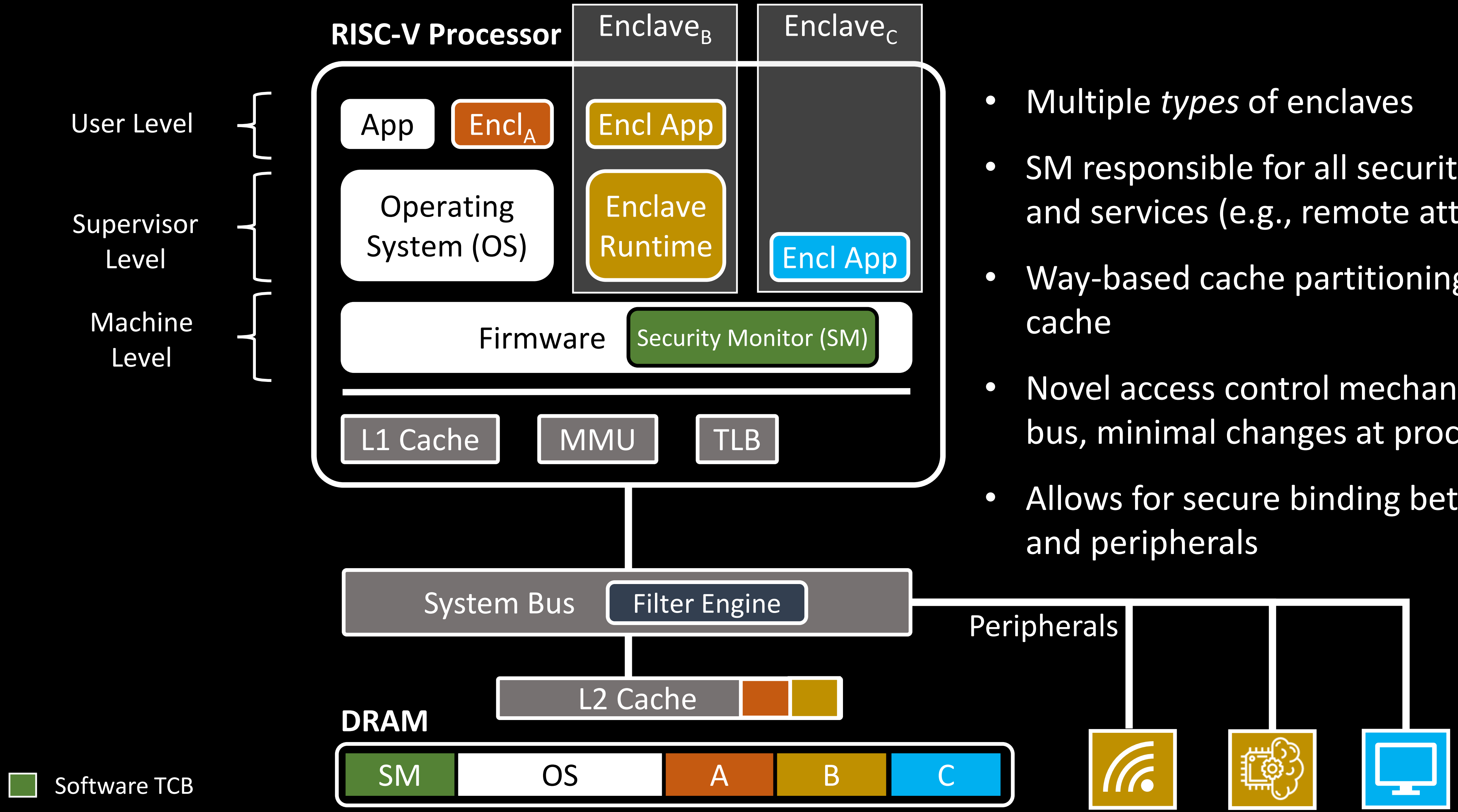


CURE: Customizable and Resilient Enclaves



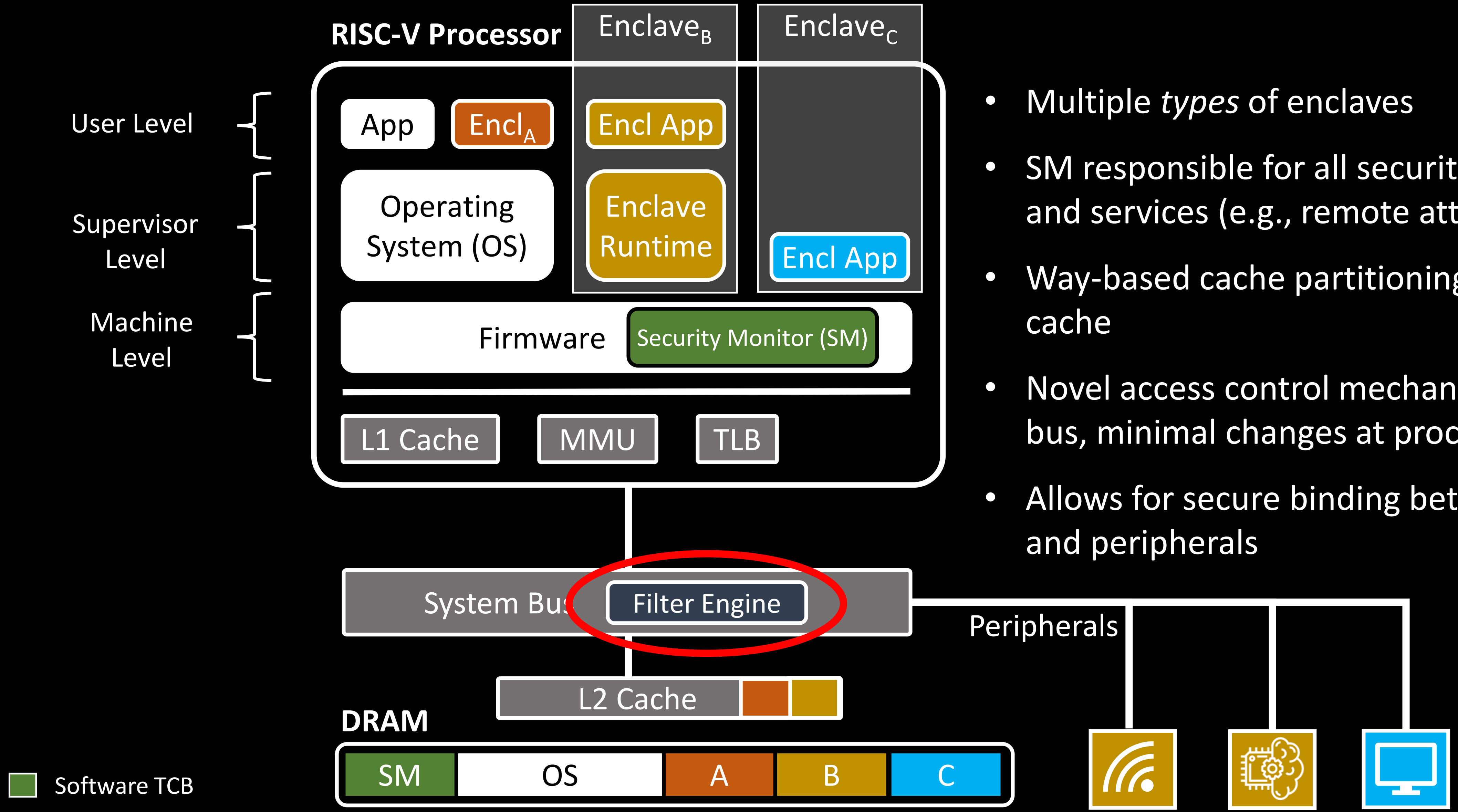
- Multiple *types* of enclaves
- SM responsible for all security-critical tasks and services (e.g., remote attestation)

CURE: Customizable and Resilient Enclaves



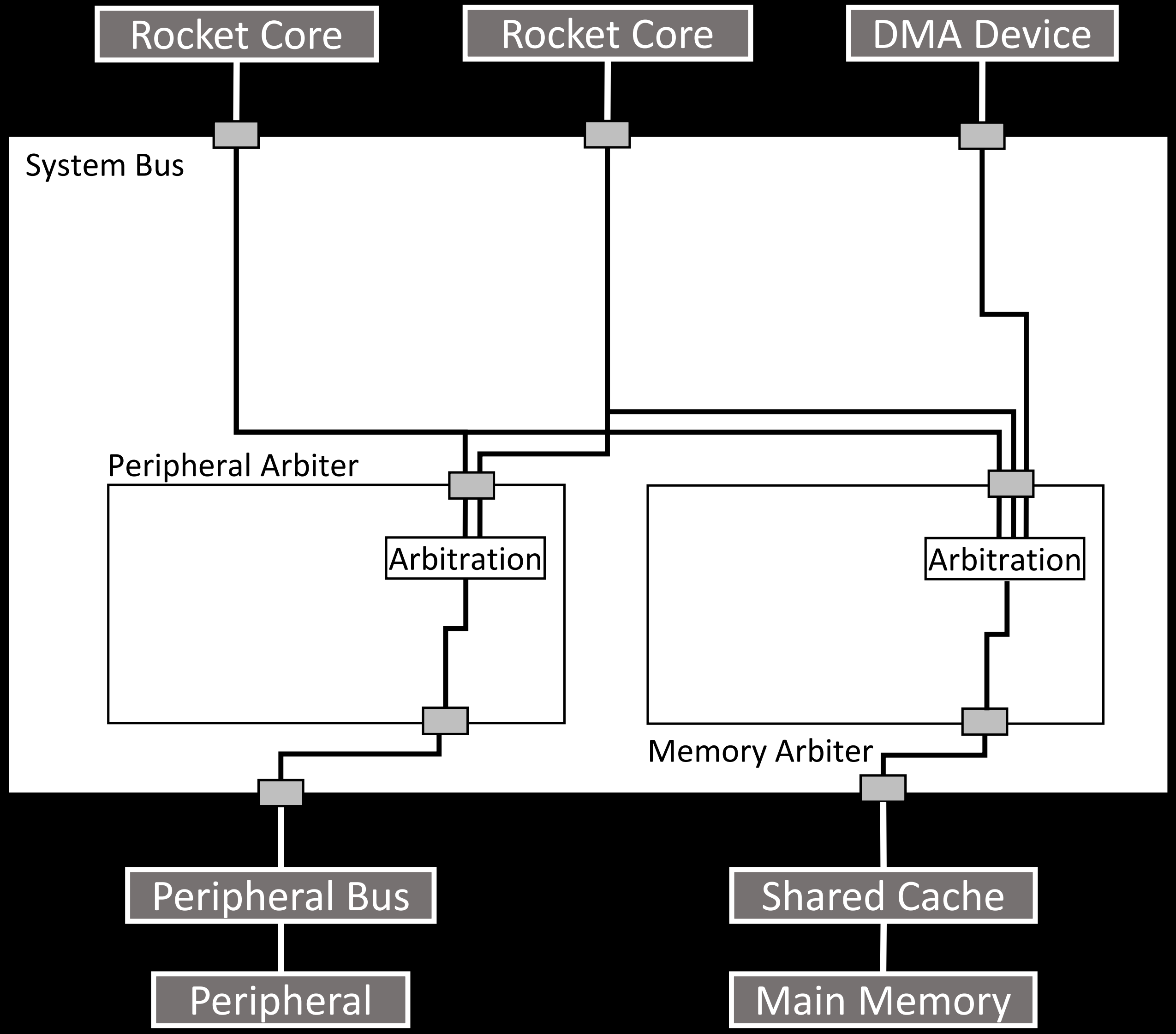
- Multiple *types* of enclaves
- SM responsible for all security-critical tasks and services (e.g., remote attestation)
- Way-based cache partitioning on shared L2 cache
- Novel access control mechanism on system bus, minimal changes at processor
- Allows for secure binding between enclaves and peripherals

CURE: Customizable and Resilient Enclaves



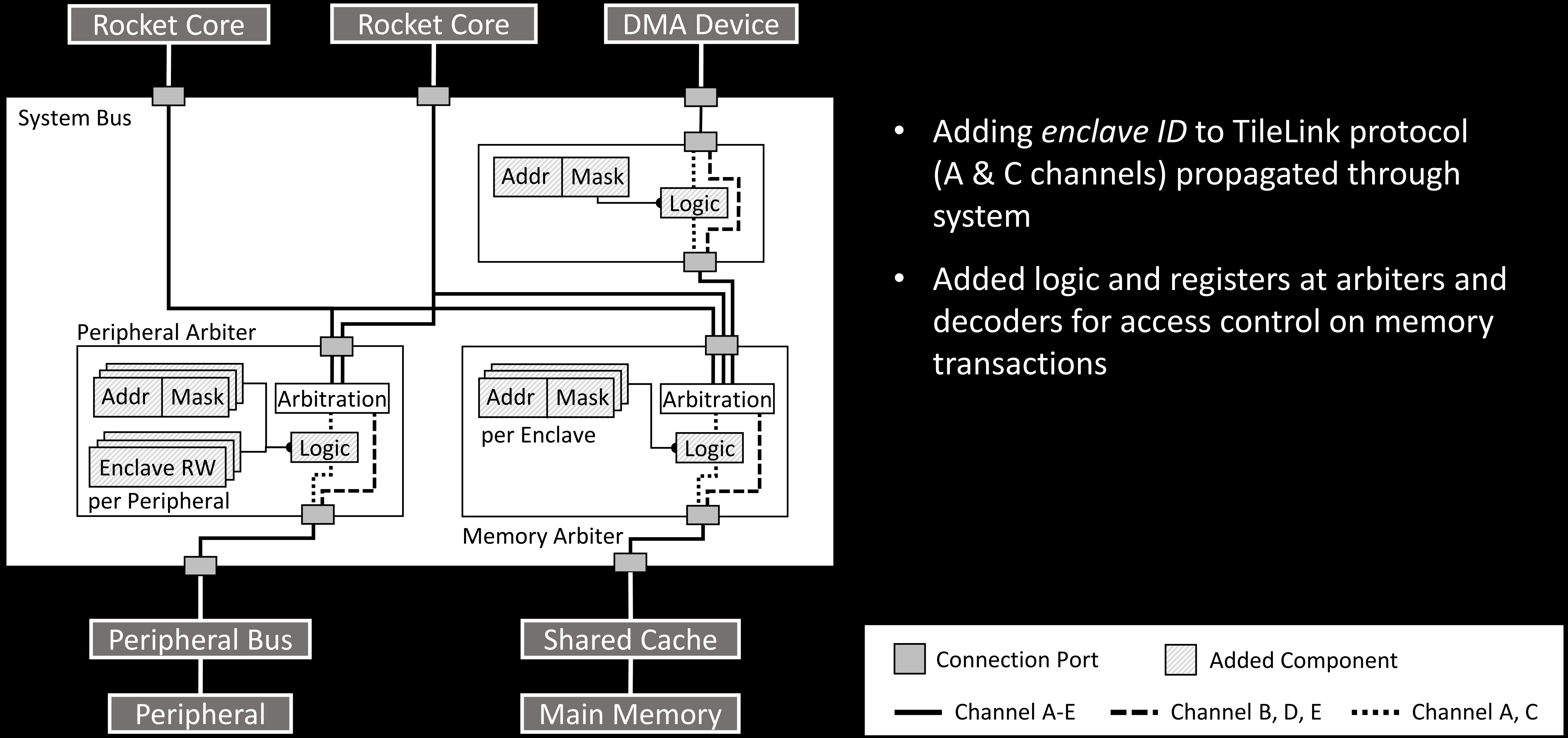
- Multiple *types* of enclaves
- SM responsible for all security-critical tasks and services (e.g., remote attestation)
- Way-based cache partitioning on shared L2 cache
- Novel access control mechanism on system bus, minimal changes at processor
- Allows for secure binding between enclaves and peripherals

Details on System Bus Access Control



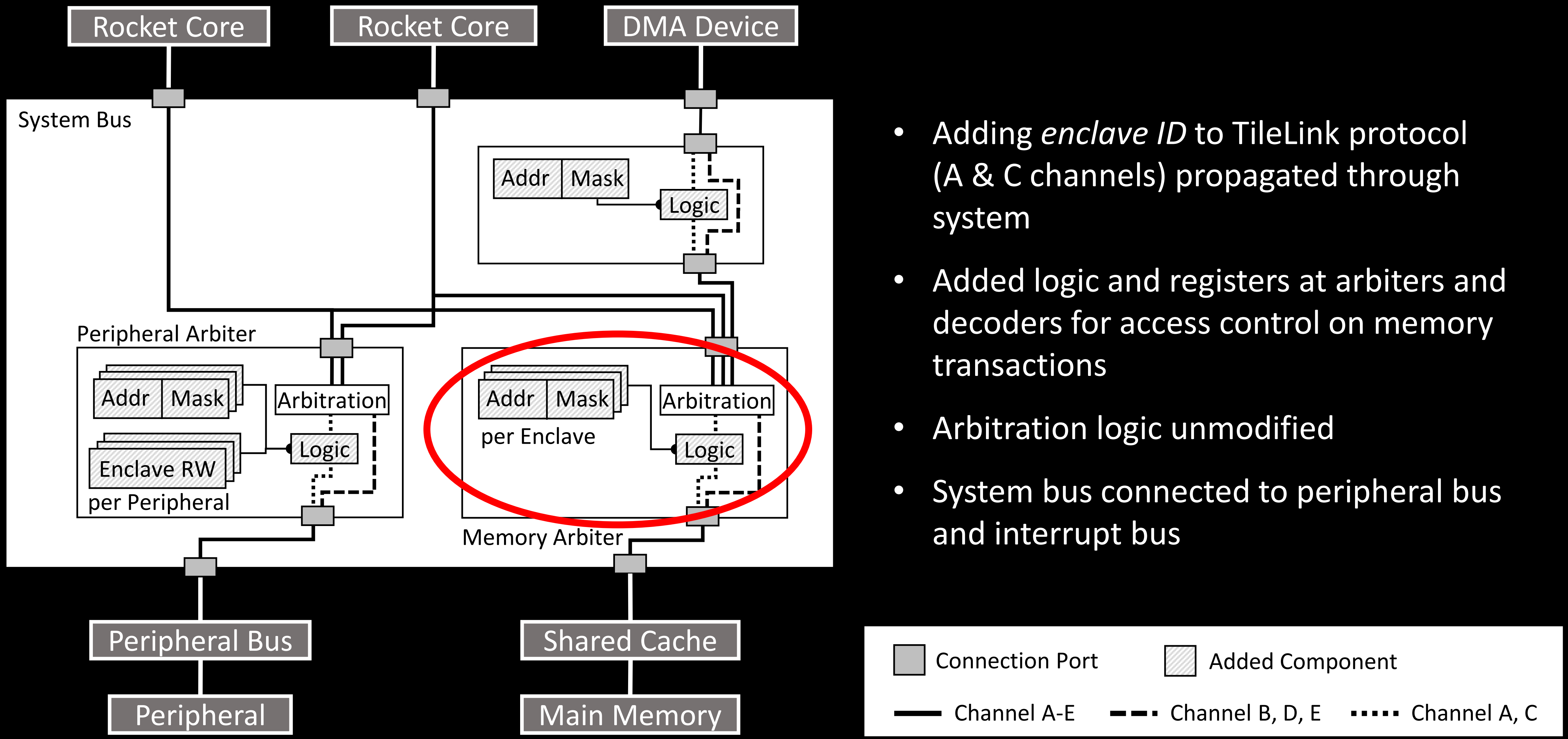
- Adding *enclave ID* to TileLink protocol (A & C channels) propagated through system

Details on System Bus Access Control



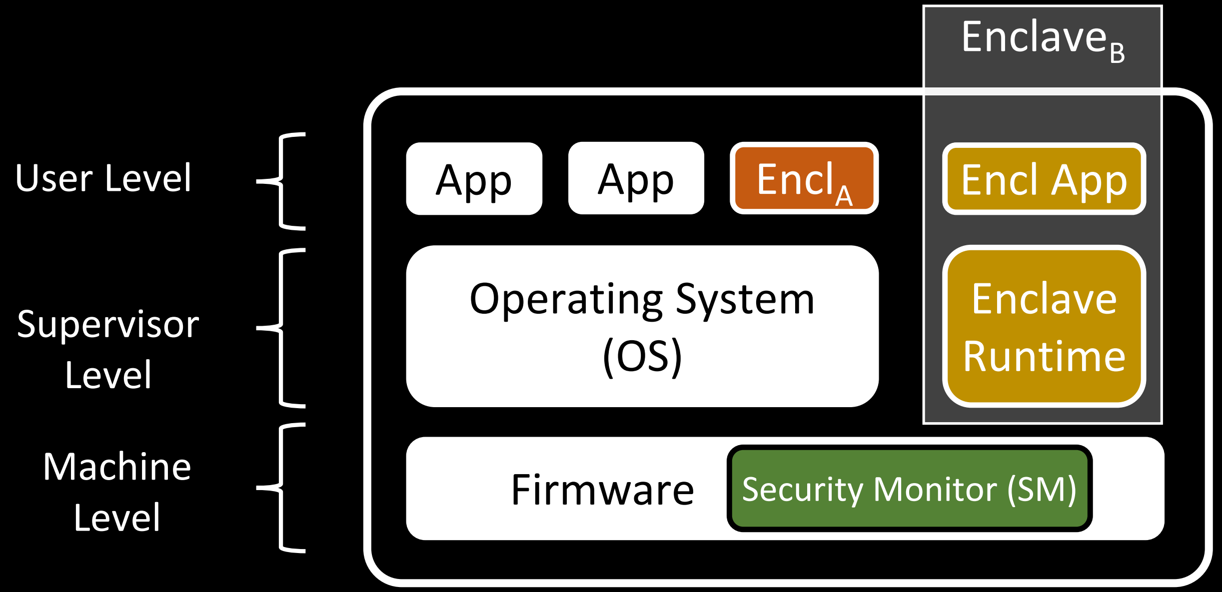
- Adding *enclave ID* to TileLink protocol (A & C channels) propagated through system
- Added logic and registers at arbiters and decoders for access control on memory transactions

Details on System Bus Access Control



- Adding *enclave ID* to TileLink protocol (A & C channels) propagated through system
- Added logic and registers at arbiters and decoders for access control on memory transactions
- Arbitration logic unmodified
- System bus connected to peripheral bus and interrupt bus

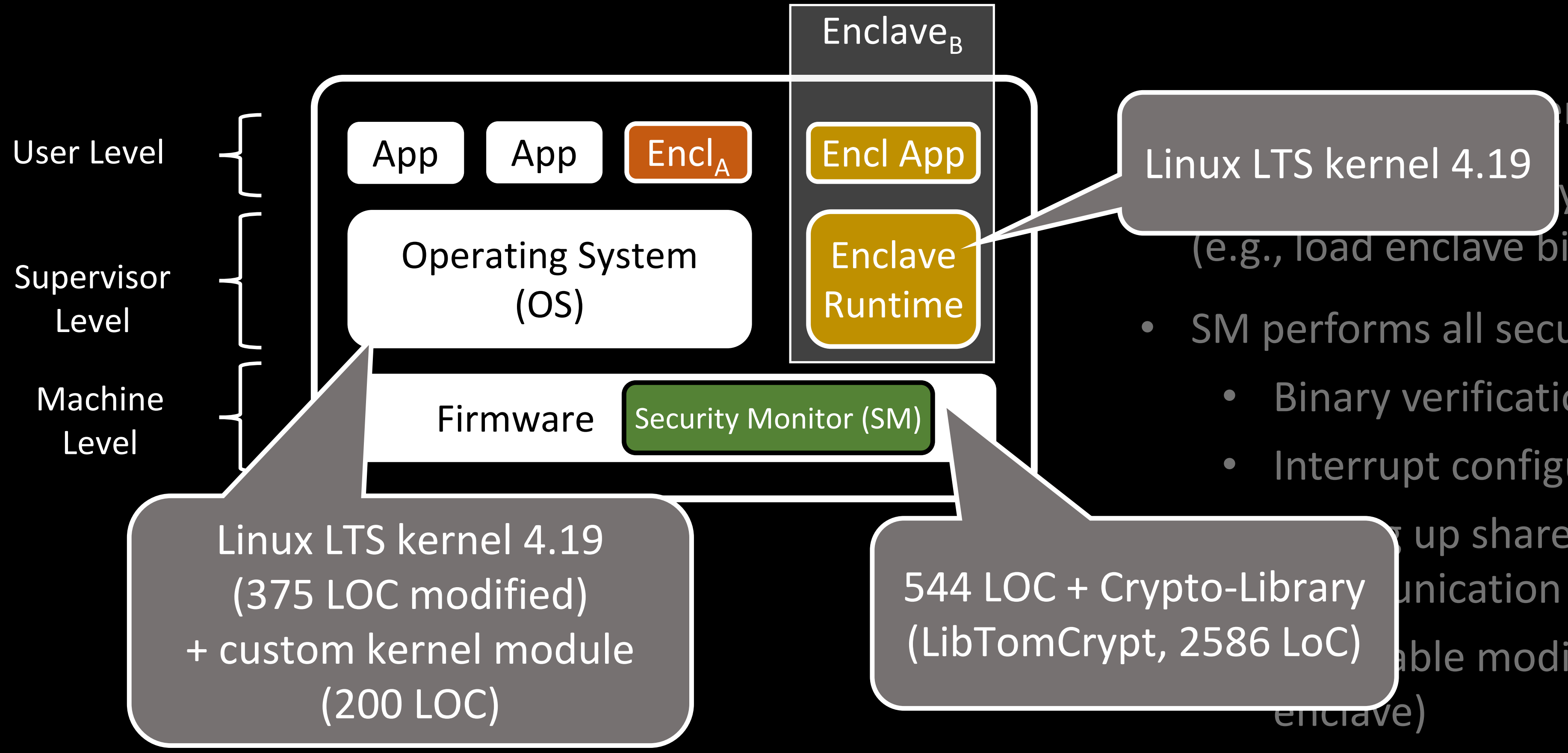
Details on Software Components



- Enclave setup triggered by OS
- OS performs security uncritical steps (e.g., load enclave binary)
- SM performs all security critical steps
  - Binary verification
  - Interrupt configuration
  - Setting up shared memory for communication
  - Page table modification (user-space enclave)

Software TCB

Details on Software Components



…red by OS

…y uncritical steps (e.g., load enclave binary)

- SM performs all security critical steps
  - Binary verification
  - Interrupt configuration
  - … up shared memory for …unication
  - …able modification (user-space enclave)

■ Software TCB

Conclusion

- CURE successfully tackles identified challenges
- Security:
  - Keep all side-channel sensitive data structures inside enclave (page tables, interrupt handlers)
  - Dynamic way-based cache partitioning
- Functionality:
  - New access control mechanism on system bus enables enclave-to-peripheral binding
- Configurability:
  - Provides multiple types of enclaves
- CURE offers many possibilities for further development
  (e.g., VM enclaves, new side-channel resilient cache architectures)

Questions ?

emmanuel.stapf@trust.tu-darmstadt.de

References 1/2

1. V. Costan, I. Lebedev, and S. Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In USENIX Security, 2016.
2. F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. In HASP. ACM, 2013.
3. A. Ahmad, B. Joe, Y. Xiao, Y. Zhang, I. Shin, and B. Lee. Obfuscuro: A commodity obfuscation engine on intel sgx. In NDSS, 2019.
4. A. Ahmad, K. Kim, M. I. Sarfaraz, and B. Lee. Obliviate: A data oblivious filesystem for intel sgx. In NDSS, 2018.
5. F. Brasser, S. Capkun, A. Dmitrienko, T. Frassetto, K. Kostiainen, and A. Sadeghi. Dr. sgx: automated and adjustable side-channel protection for sgx using data location randomization. In ACSAC, pages 788–800, 2019.
6. S. Chen, X. Zhang, M. K. Reiter, and Y. Zhang. Detecting privileged side-channel attacks in shielded execution with déjá vu. In Asia CCS, pages 7–18. ACM, 2017.
7. O. Oleksenko, B. Trach, R. Krahn, M. Silberstein, and C. Fetzer. Varys: Protecting sgx enclaves from practical side-channel attacks. In USENIX ATC, 2018.
8. M. Shih, S. Lee, T. Kim, and M. Peinado. T-sgx: Eradicating controlled-channel attacks against enclave programs. In NDSS, 2017.
9. ARM Limited. Security technology: building a secure system using TrustZone technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, 2008.

References 2/2

10. F. Brasser, D. Gens, P. Jauernig, A. Sadeghi, and E. Stapf. Sanctuary: Arming trustzone with user-space enclaves. In NDSS, 2019.
11. D. Lee, D. Kohlbrenner, S. Shinde, D. Song, and K. Asanovi´c. Keystone: An open framework for architecting trusted execution environments. EuroSys, 2019.