Extending RISC-V Keystone to Include Efficient Secure Memory

- **Zach Moolman**
- Tamara Silbergleit Lehman

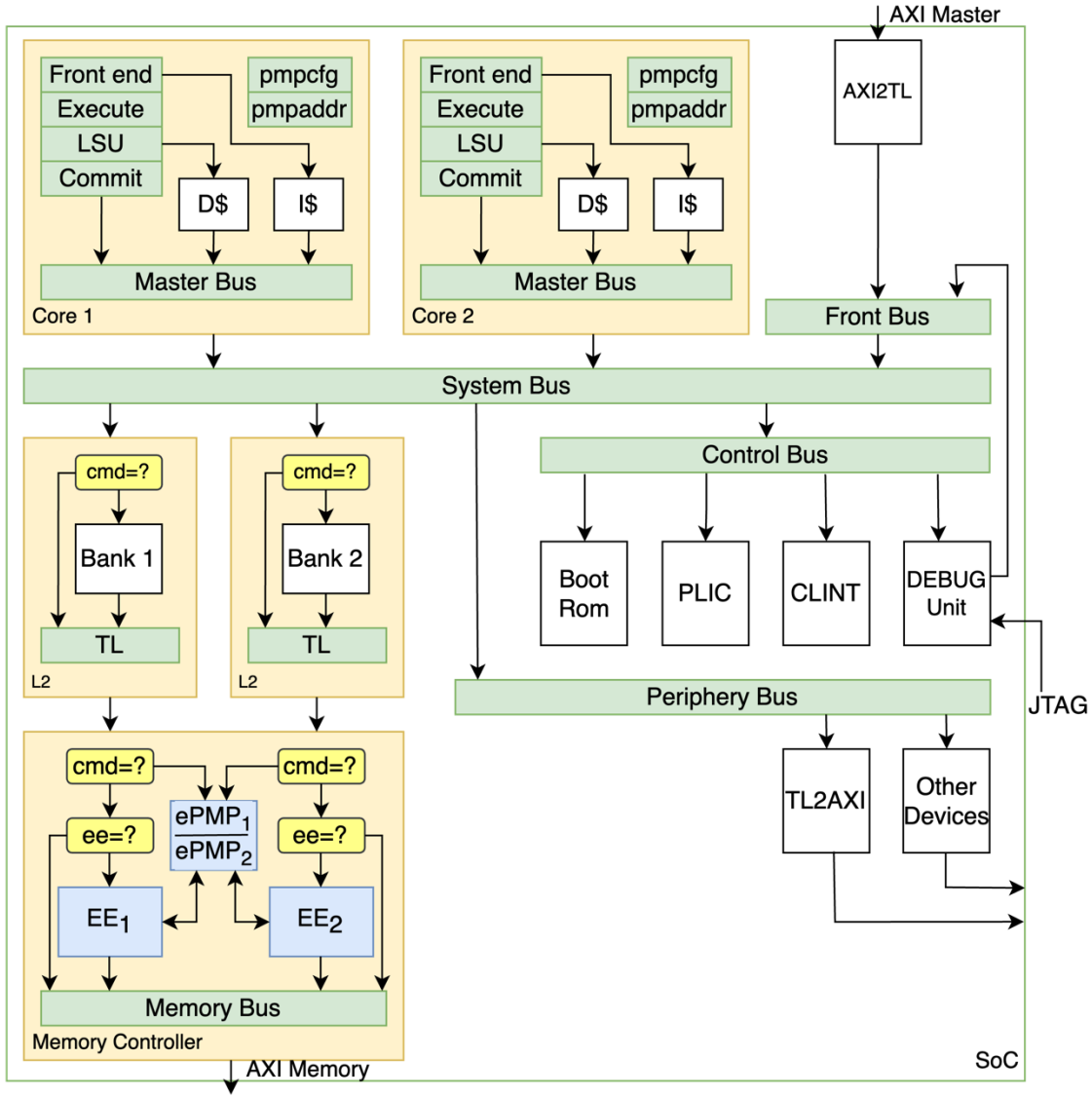
Electrical, Computer & Energy Engineering
College of Engineering and Applied Science

University of Colorado **Boulder**

Why are we here?

To build micro-architecture systems where security is a first-class citizen.

Secure Memory in Hardware



GHOST

Background

Multi discipline project
NSF funded
One of GHOST's goals is to operate securely through public 5G network

GHOST

The problem

Using network – exposing your activities
Data are encrypted – information can be derived
Network controlled by adversary

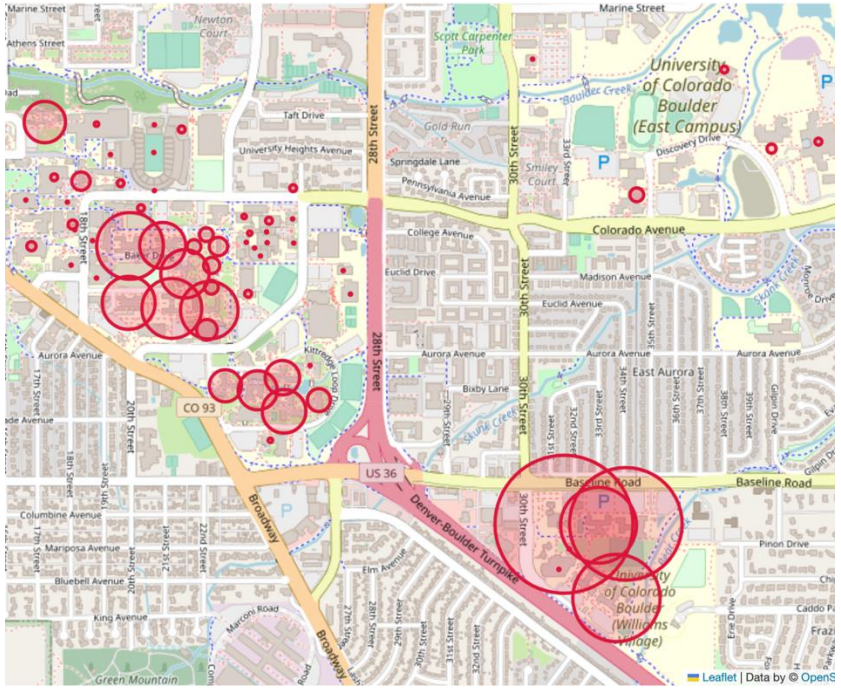
GHOST

Features

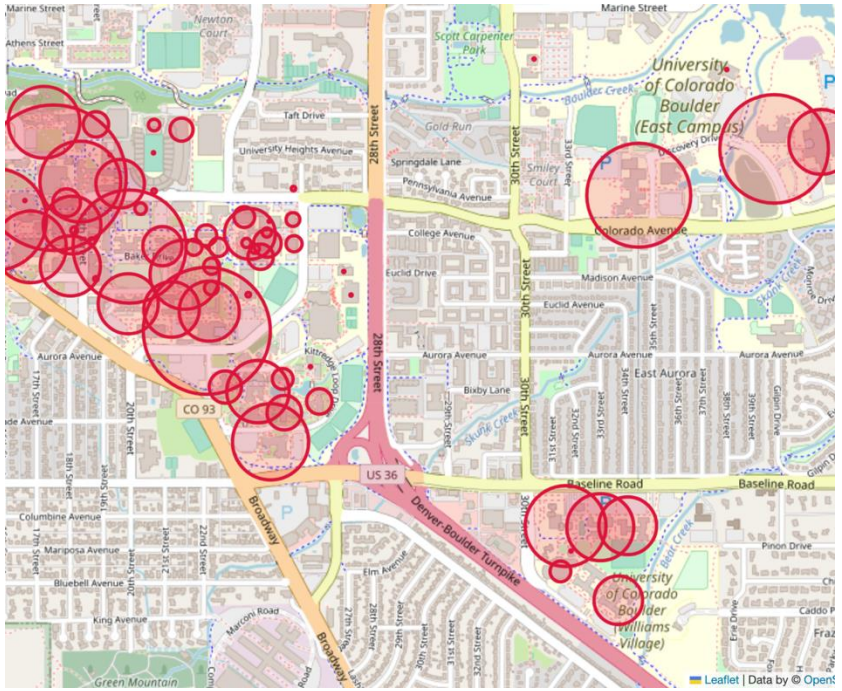
Activity shaping
Sim swapping

GHOST

Exposing your activities



Network device count at 6:58 am



Network device count at 12:40 pm

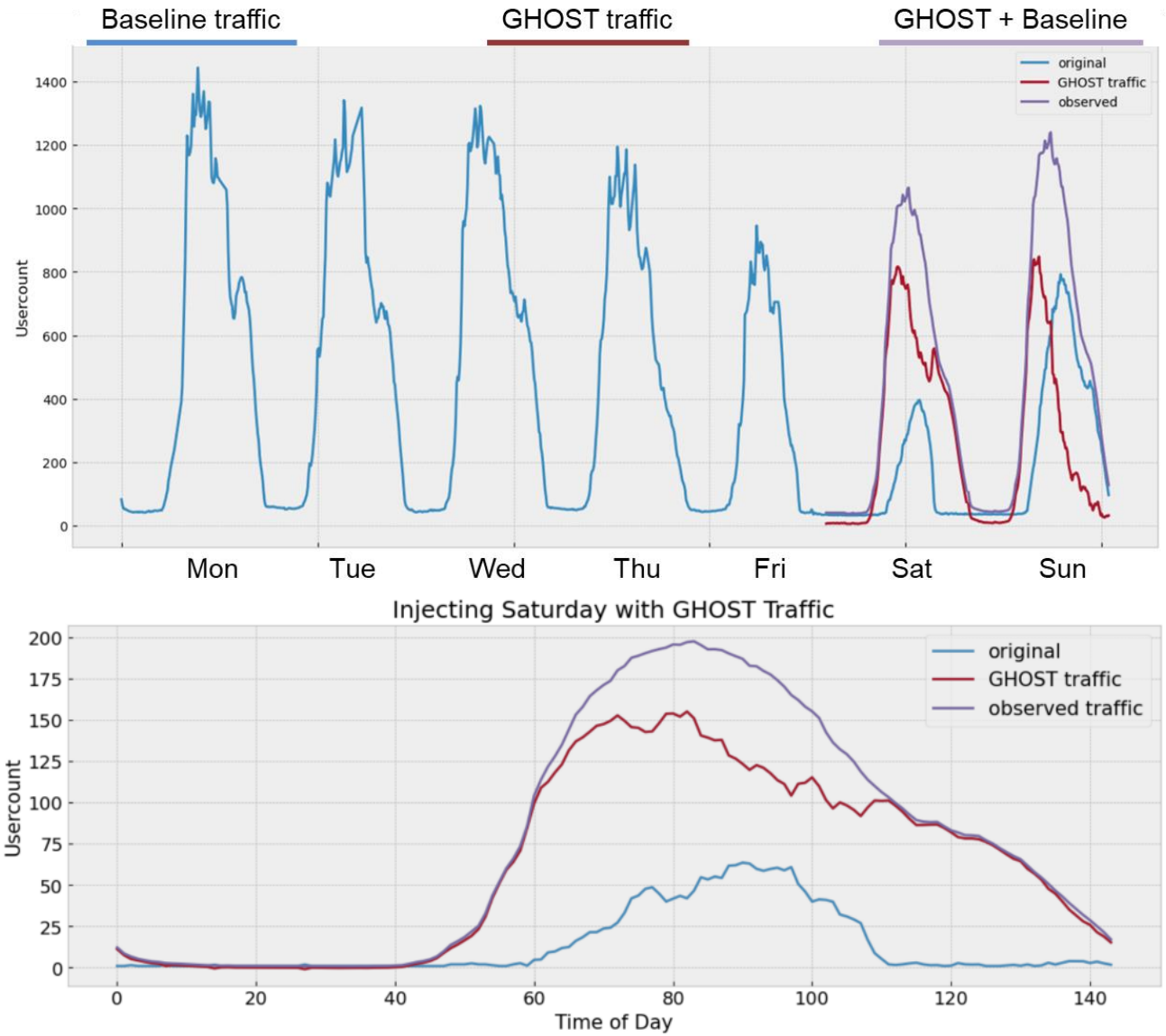
Tschimben, S., Bates, I., Curry, J. H., Gremban, K. D., & Siegel, A. (2023, October). Modeling and Generation of Realistic Network Activity. In MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM) (pp. 761-766). IEEE.

GHOST

Hiding in Plain Sight

Weekend network activity look like weekday activity

NMF model changes the observed pattern of network activity

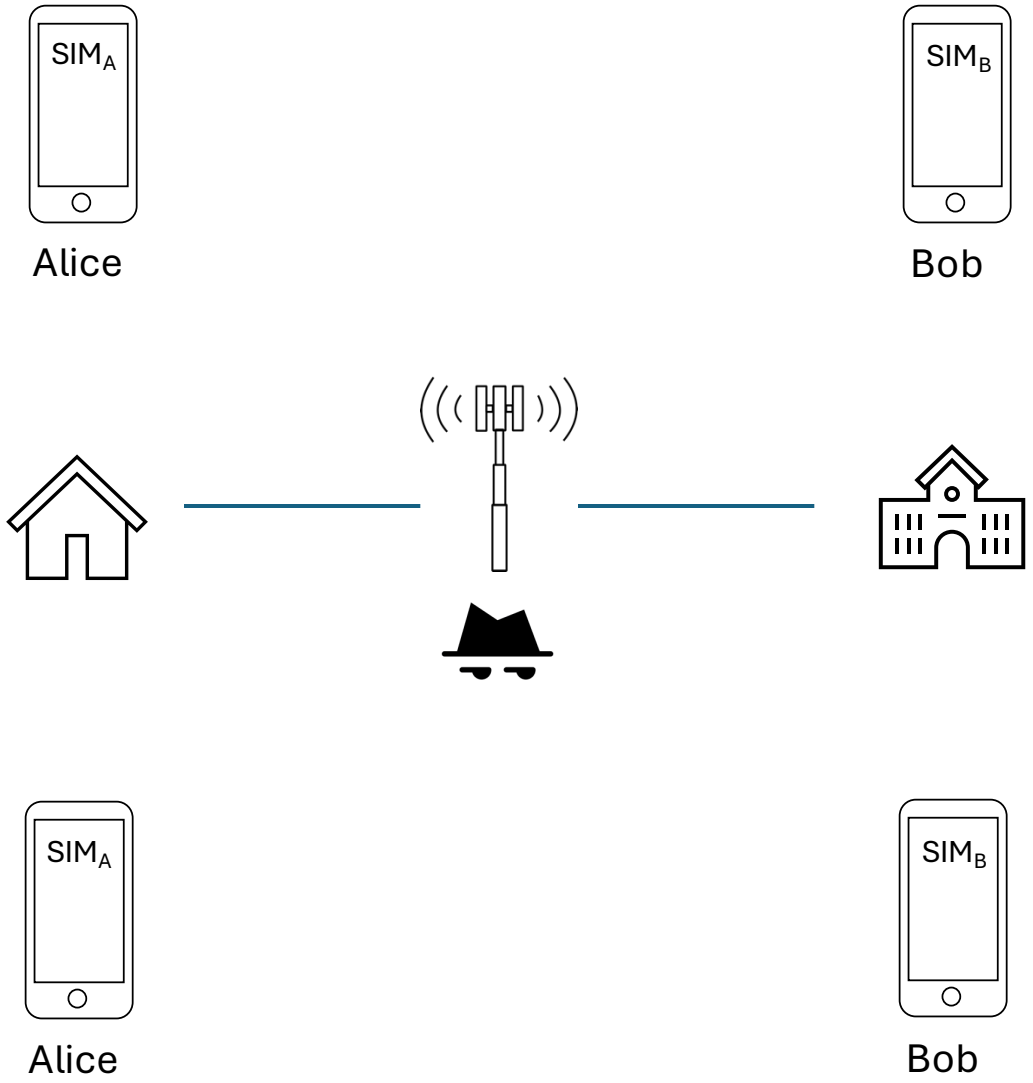


Tschimben, S., Bates, I., Curry, J. H., Gremban, K. D., & Siegel, A. (2023, October). Modeling and Generation of Realistic Network Activity. In MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM) (pp. 761-766). IEEE.

GHOST

Sim Swapping

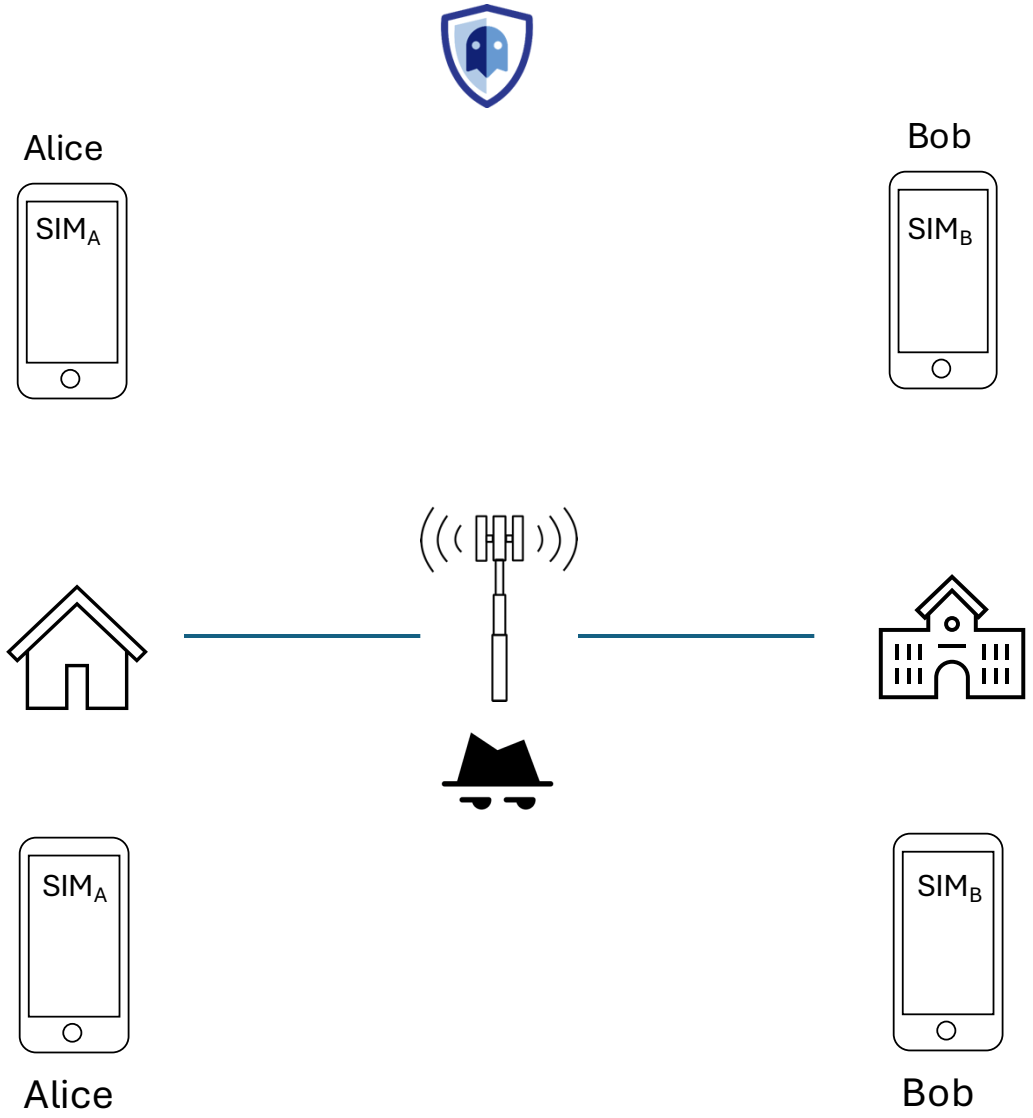
- Normal behavior
- Observer



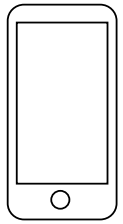
GHOST

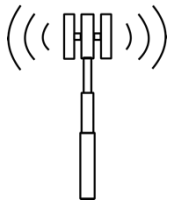
Sim Swapping

- Normal behavior
- Observer
- GHOST Enabled
- Observer



GHOST





Device Security

Stolen
Remotely Access
Require additional security

- Isolation (TEE)
- Confidentiality and Integrity (Encryption)

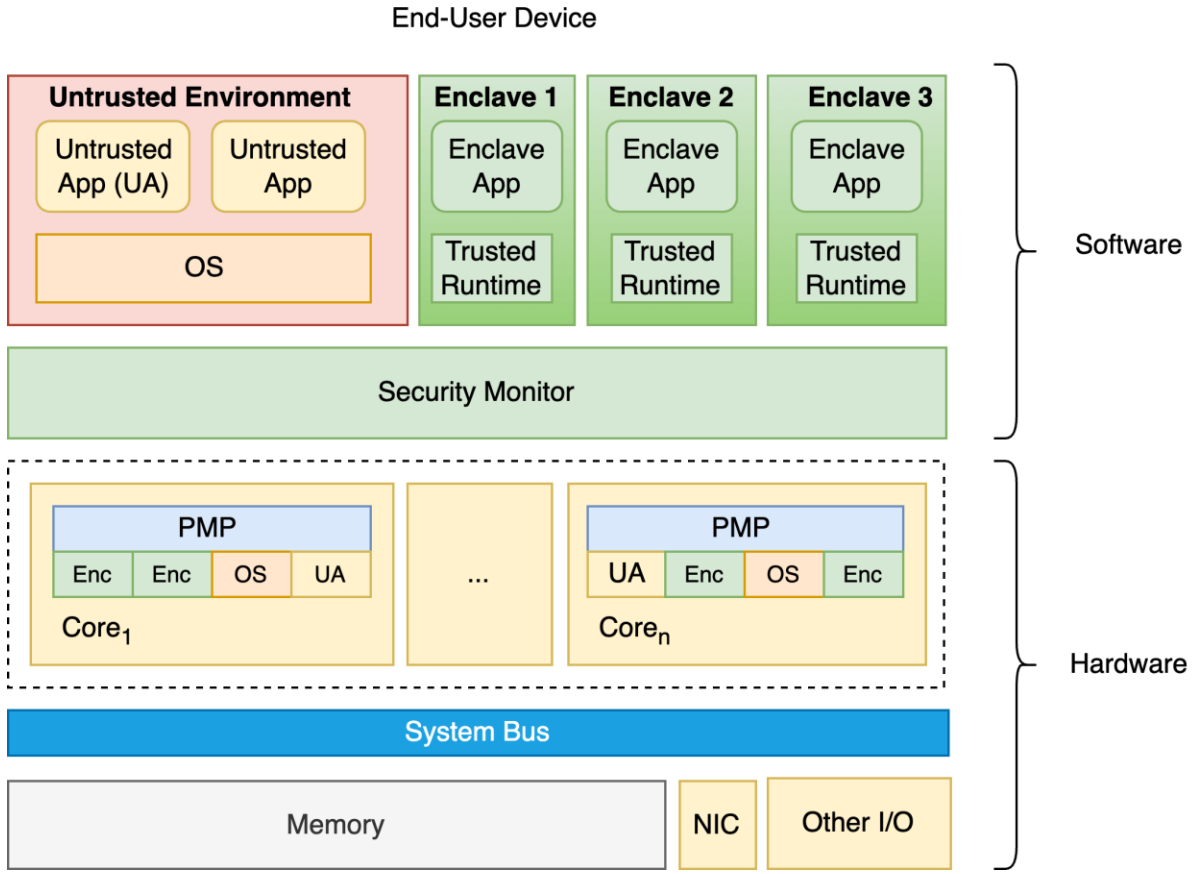
Trusted Execution Environments (TEEs)

- What is a TEE?
  - Isolation
  - Privileged Software (OS)
  - Attestation Hardware & Software
  - Trust Chip/SoC Boundary
- Commercial TEEs
  - Intel SGX – Large TCB
  - Arm TrustZone – Two Worlds

Extending RISC-V Keystone To Include Efficient Secure Memory

Keystone

- ❖Dayeol Lee
- ❖Threat Model
- ❖Open Source
- ❖Custom TEE



Lee, D., Kohlbrenner, D., Shinde, S., Asanović, K., and Song, D. Keystone: Anopen framework for architecting trusted execution environments. In Proceedings of the Fifteenth European Conference on Computer Systems (2020), pp. 1–16.

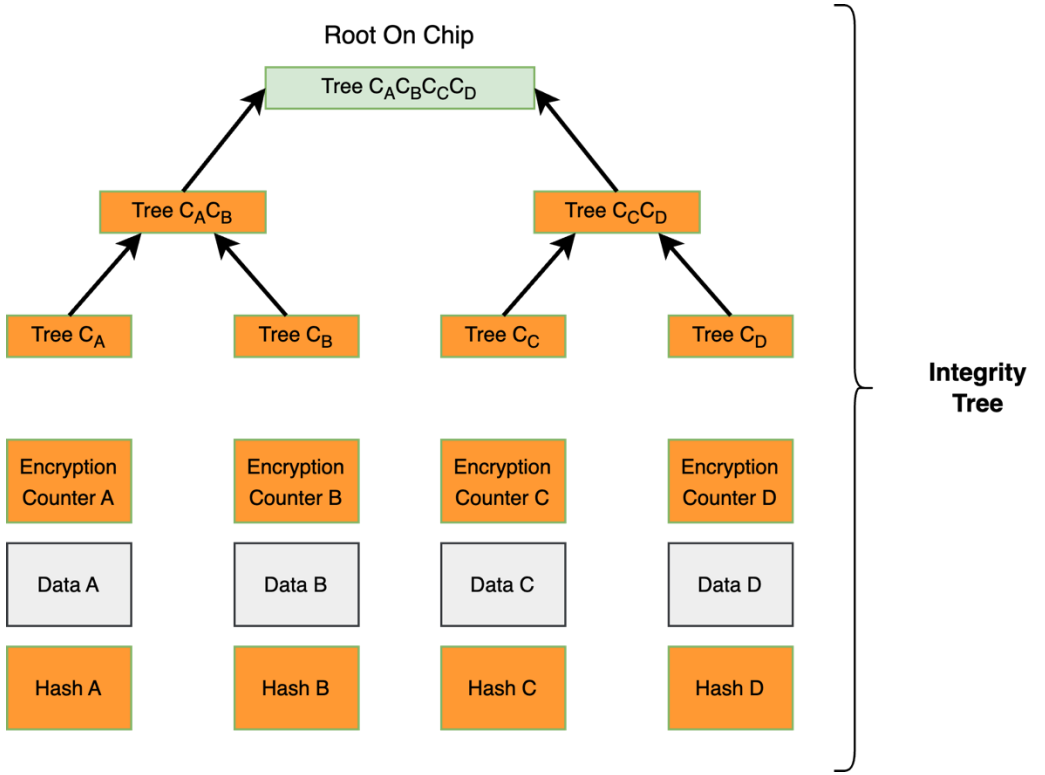
RISC-V

- Privileged Levels
  - User
  - Supervisory
  - Reserved (Hypervisor)
  - **Machine**
- Control Status Registers (CSR)
  - Physical Memory Protection (PMP)

Extending RISC-V Keystone Include Efficient Secure Memory

Secure Memory

- ❖Isolation is not enough
- ❖Confidentiality
  - Protect sensitive data
  - Counter-mode encryption
  - Encryption is parallelized with data requests
- ❖Data integrity
  - Active attacks
  - Bonsai Merkle trees



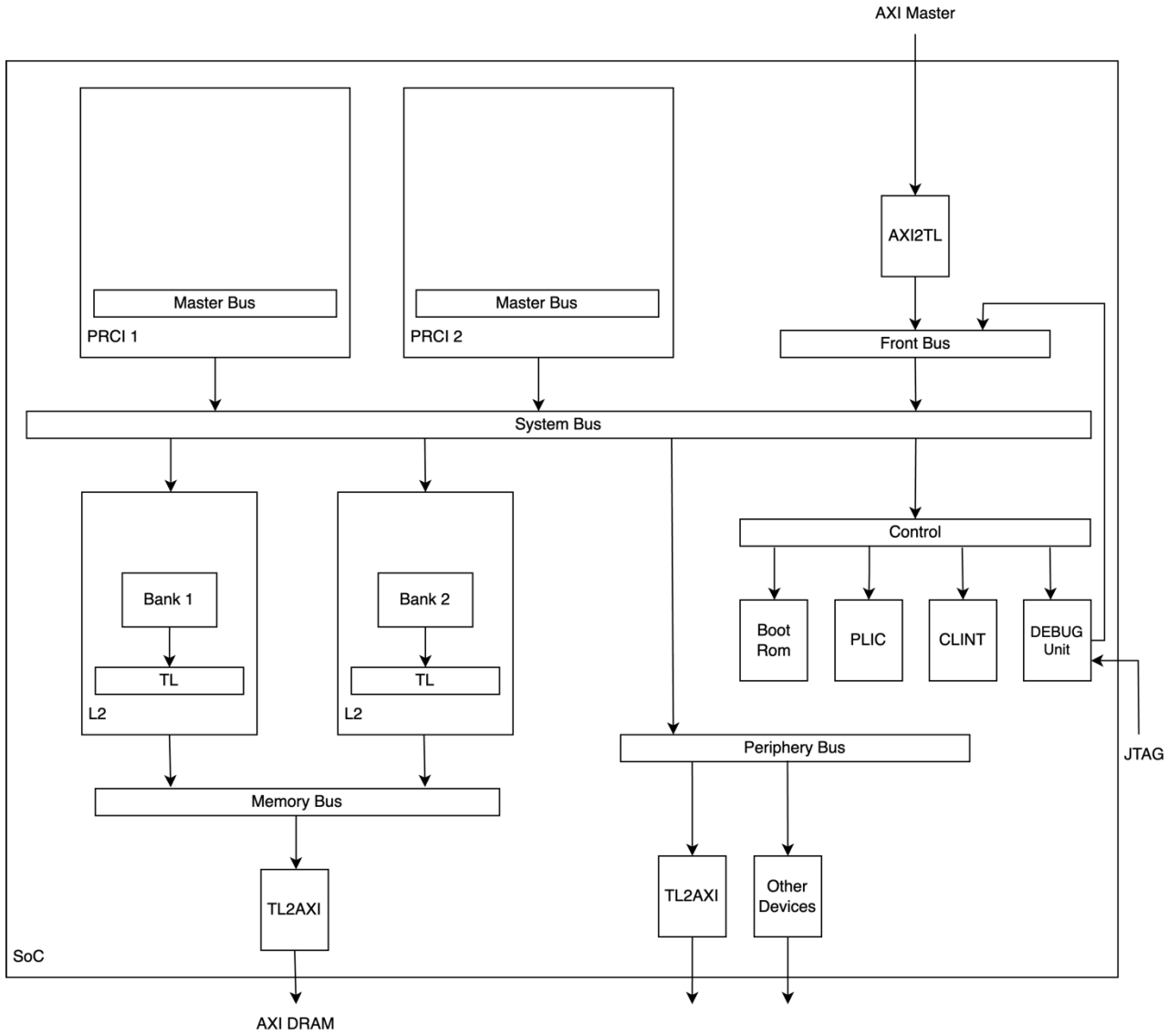
Rogers, B., Chhabra, S., Prvulovic, M., and Solihin, Y. Using address inde-pendent seed encryption and Bonsai Merkle trees to make secure processors OS- and performance-friendly. In International Symposium on Microarchitecture(MICRO) (2007)

Our Work

Extending RISC-V Keystone To Include Efficient Secure Memory

Extending RISC-V

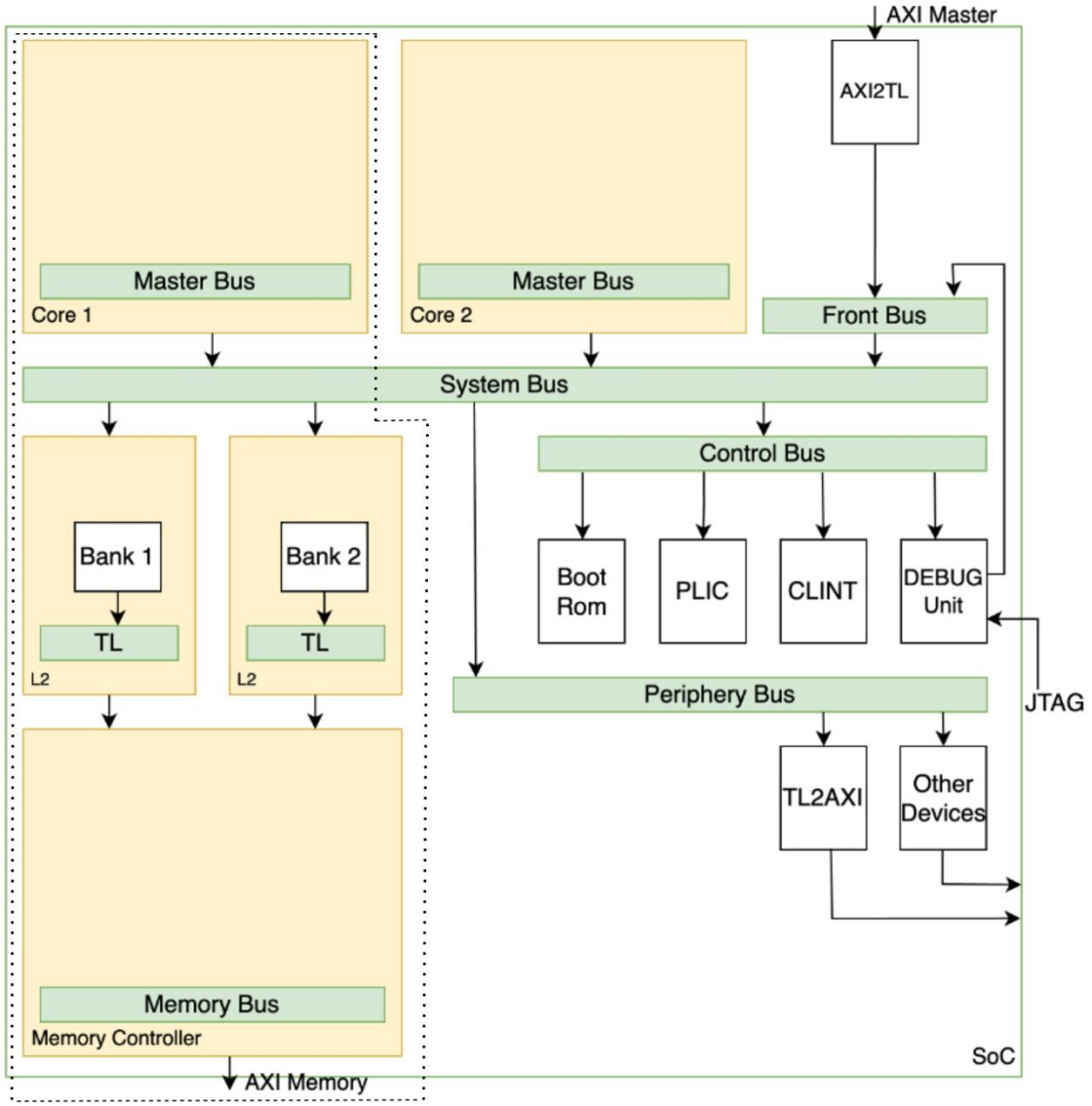
- ❖RocketChip SoC
- ❖Building Secure Memory
- ❖FPGA
- ❖Extra dimension



Extending RISC-V Keystone To Include Efficient Secure Memory

Extending RISC-V

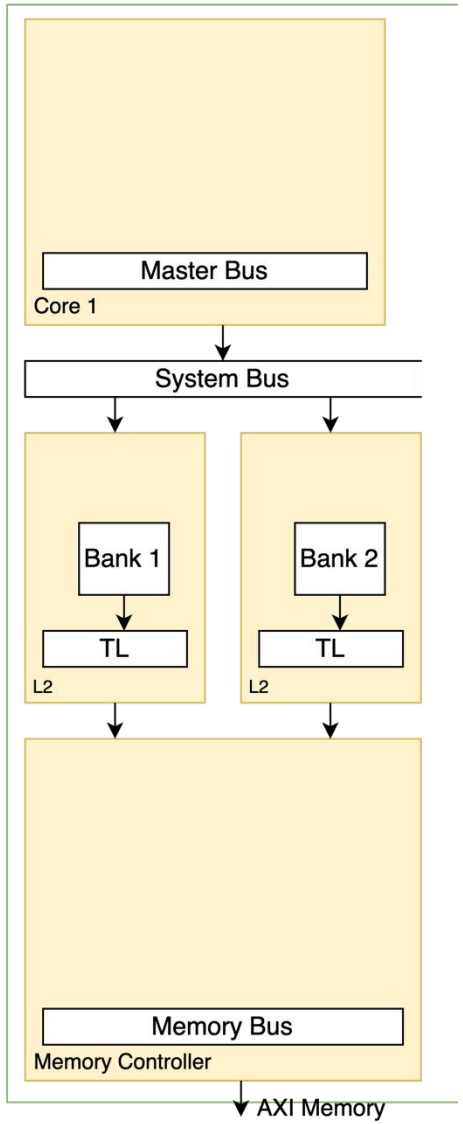
- ❖RocketChip SoC
- ❖Building Secure Memory
- ❖FPGA
- ❖Extra dimension



Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

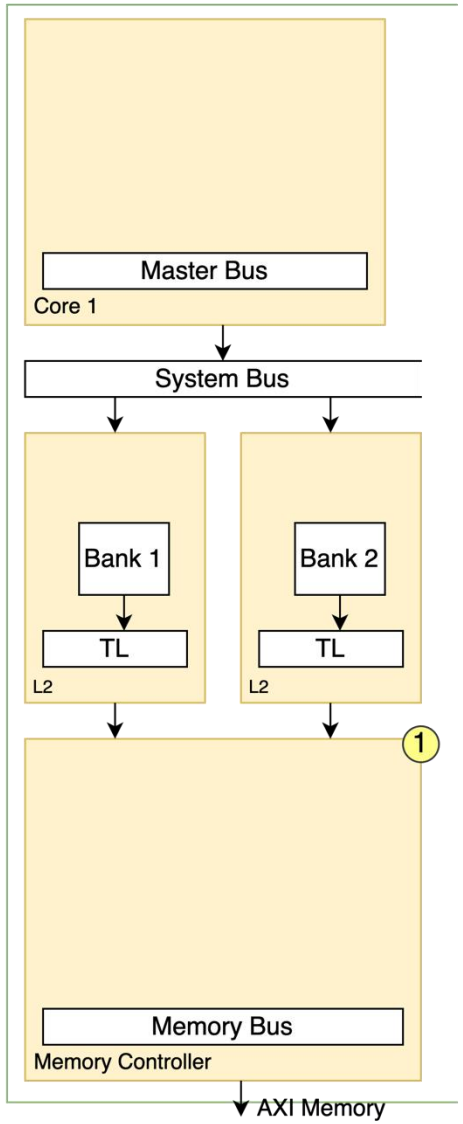
- Confidentiality
  - Memory Controller (MC)
  - Enable Encryption (ee)
  - PMPChecker
  - TileLink (TL)
  - Encryption Engine(EE)
  - Critical Path



Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Confidentiality
  - Memory Controller (MC)
  - Enable Encryption (ee)
  - PMPChecker
  - TileLink (TL)
  - Encryption Engine(EE)
  - Critical Path

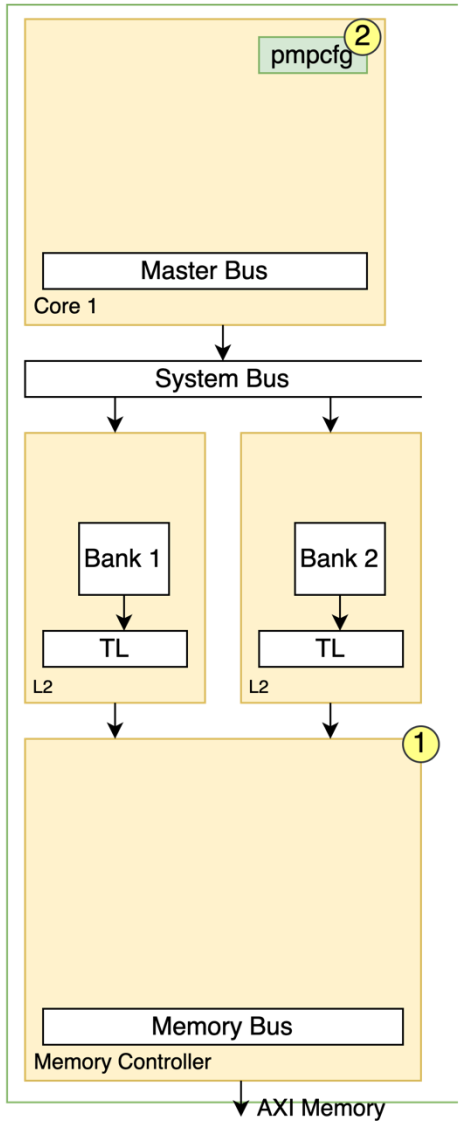


1 Change the memory bus into a memory controller

Extending RISC-V Keystone To Include Efficient Secure Memory

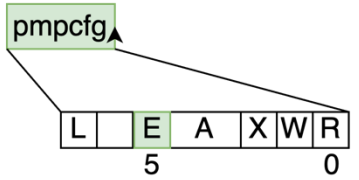
Extending SoC

- Confidentiality
  - Memory Controller (MC)
  - Enable Encryption (ee)
  - PMPChecker
  - TileLink (TL)
  - Encryption Engine(EE)
  - Critical Path



1 Change the memory bus into a memory controller

2 Add encryption bit to design

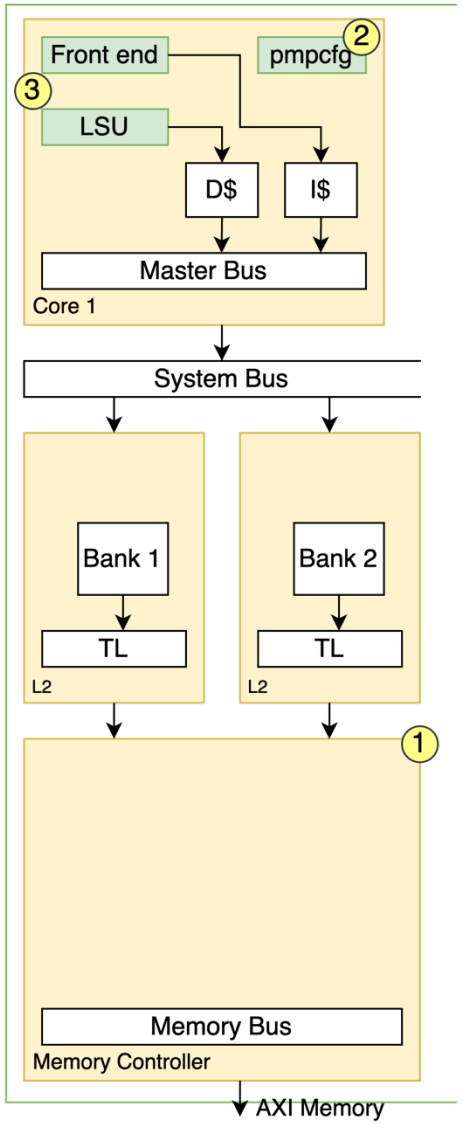


PMP configure register format

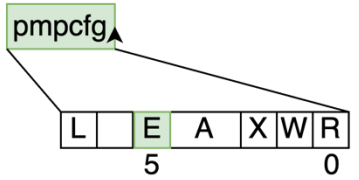
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Confidentiality
  - Memory Controller (MC)
  - Enable Encryption (ee)
  - PMPChecker
  - TileLink (TL)
  - Encryption Engine(EE)
  - Critical Path



1. Change the memory bus into a memory controller
2. Add encryption bit to design



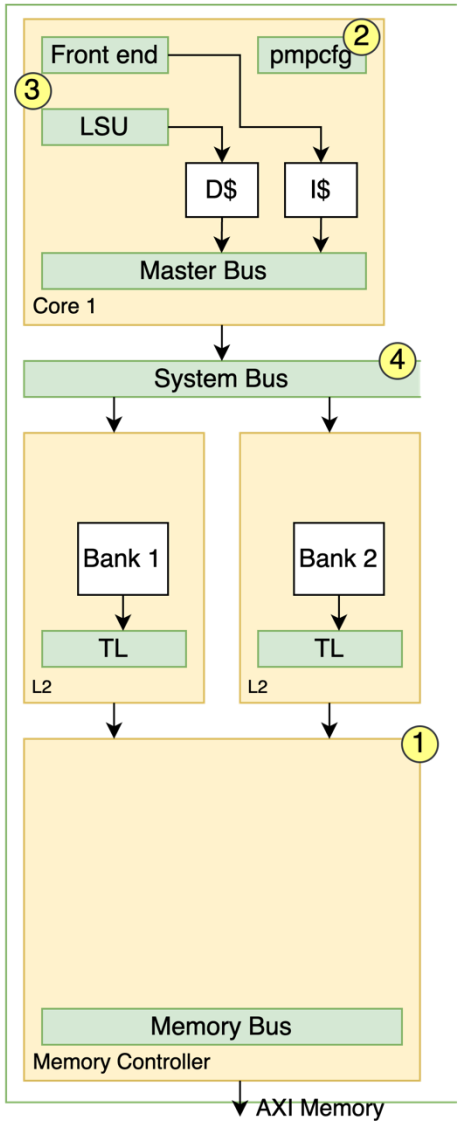
PMP configure register format

3. PMPChecker fetch encryption bit

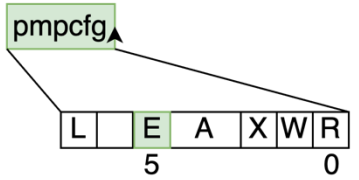
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- ❖Confidentiality
  - Memory Controller (MC)
  - Enable Encryption (ee)
  - PMPChecker
  - TileLink (TL)
  - Encryption Engine(EE)
  - Critical Path



1. Change the memory bus into a memory controller
2. Add encryption bit to design



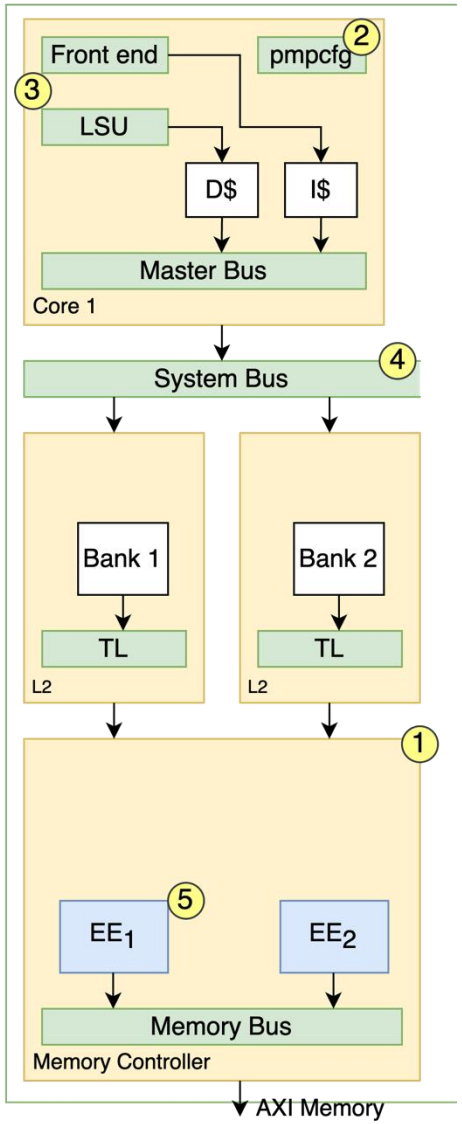
PMP configure register format

3. PMPChecker fetch encryption bit
4. Update TileLink protocol to include encryption signal

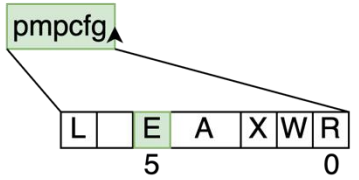
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Confidentiality
  - Memory Controller (MC)
  - Enable Encryption (ee)
  - PMPChecker
  - TileLink (TL)
  - Encryption Engine(EE)
  - Critical Path



1. Change the memory bus into a memory controller
2. Add encryption bit to design



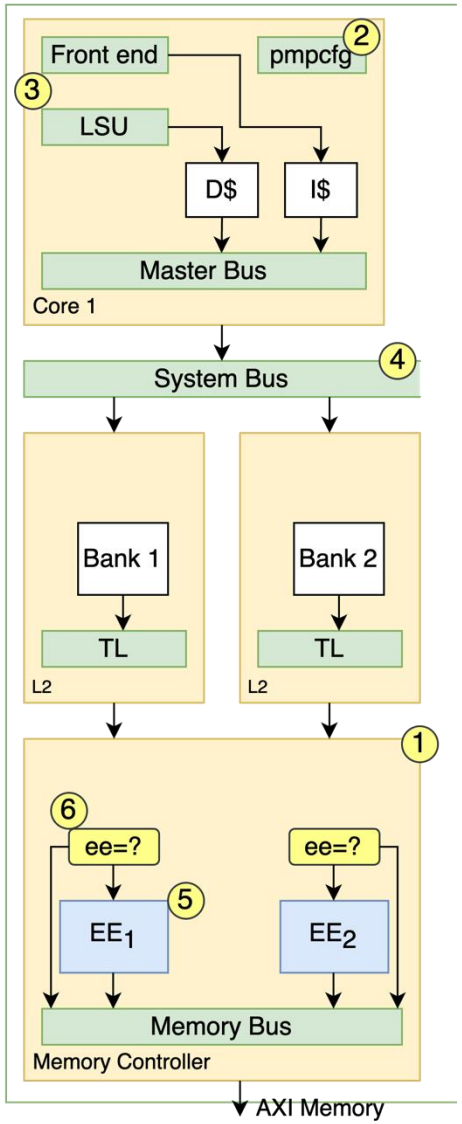
PMP configure register format

3. PMPChecker fetch encryption bit
4. Update TileLink protocol to include encryption signal
5. Add encryption engine to memory controller

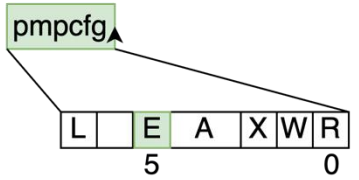
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Confidentiality
  - Memory Controller (MC)
  - Enable Encryption (ee)
  - PMPChecker
  - TileLink (TL)
  - Encryption Engine(EE)
  - Critical Path



1. Change the memory bus into a memory controller
2. Add encryption bit to design



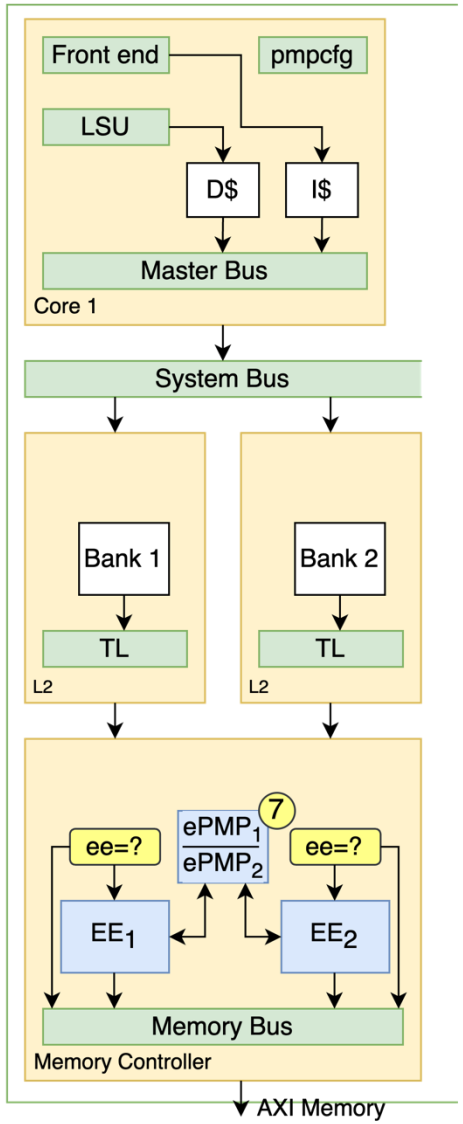
PMP configure register format

3. PMPChecker fetch encryption bit
4. Update TileLink protocol to include encryption signal
5. Add encryption engine to memory controller
6. Add one LUT to critical path

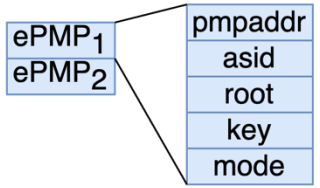
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Integrity



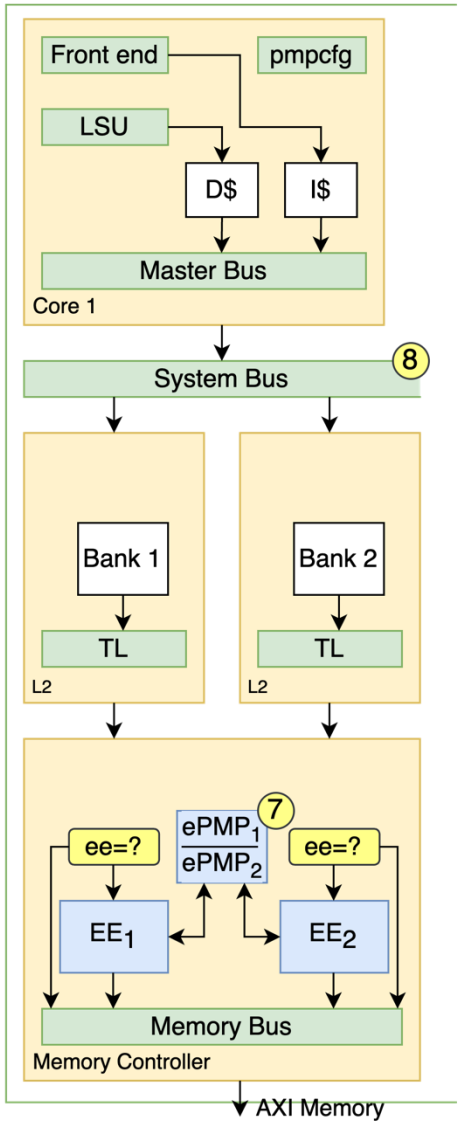
7 Add ePMP registers to hold state of integrity tree with it respective PMP entry



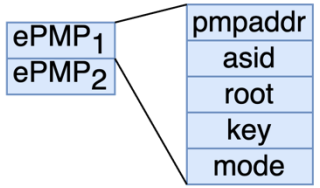
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Integrity



7 Add ePMP registers to hold state of integrity tree with it respective PMP entry

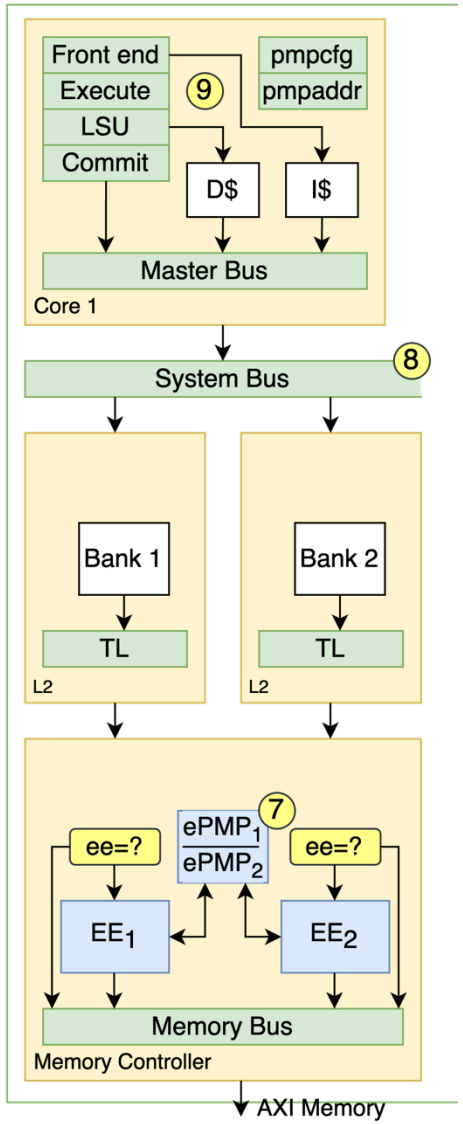


8 Update TileLink protocol to support new command

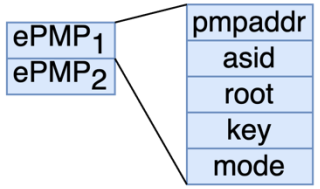
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Integrity



7 Add ePMP registers to hold state of integrity tree with it respective PMP entry



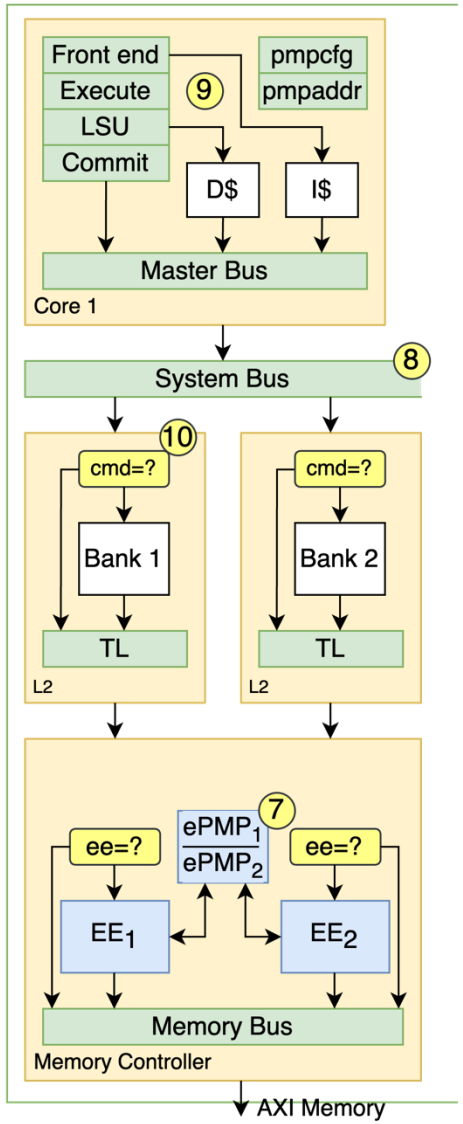
8 Update TileLink protocol to support new command

9 CSRW instruction writes to pmpaddr and to ePMP control registers

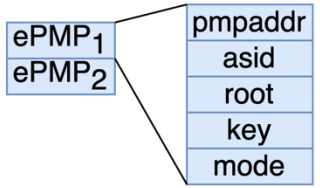
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

❖Integrity



7 Add ePMP registers to hold state of integrity tree with it respective PMP entry



8 Update TileLink protocol to support new command

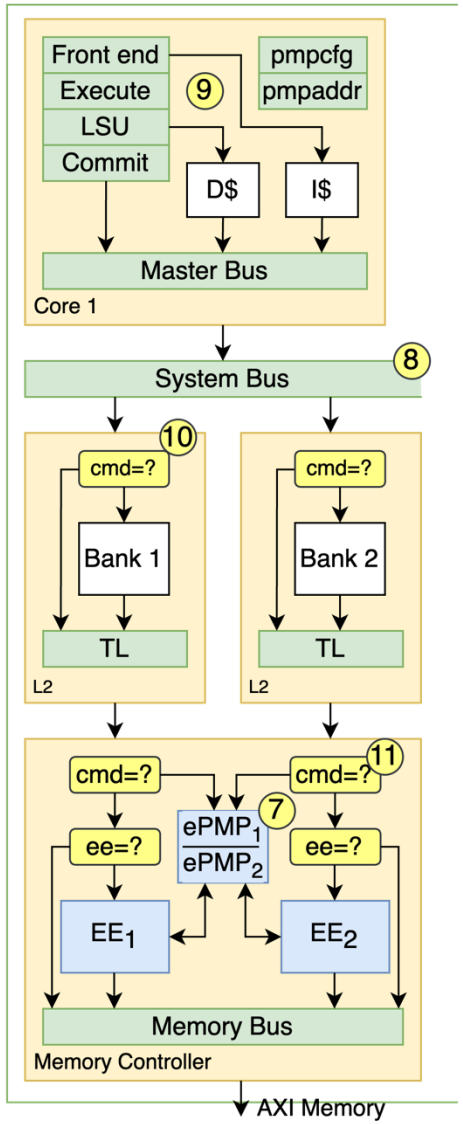
9 CSRW instruction writes to pmpaddr and to ePMP control registers

10 Add LUT to L2 Cache to either handle normal data request or forward request to Memory contorller

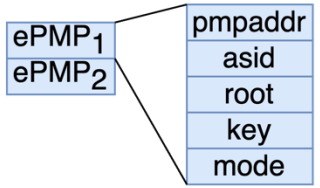
Extending RISC-V Keystone To Include Efficient Secure Memory

Extending SoC

- Integrity



7 Add ePMP registers to hold state of integrity tree with it respective PMP entry



8 Update TileLink protocol to support new command

9 CSRW instruction writes to pmpaddr and to ePMP control registers

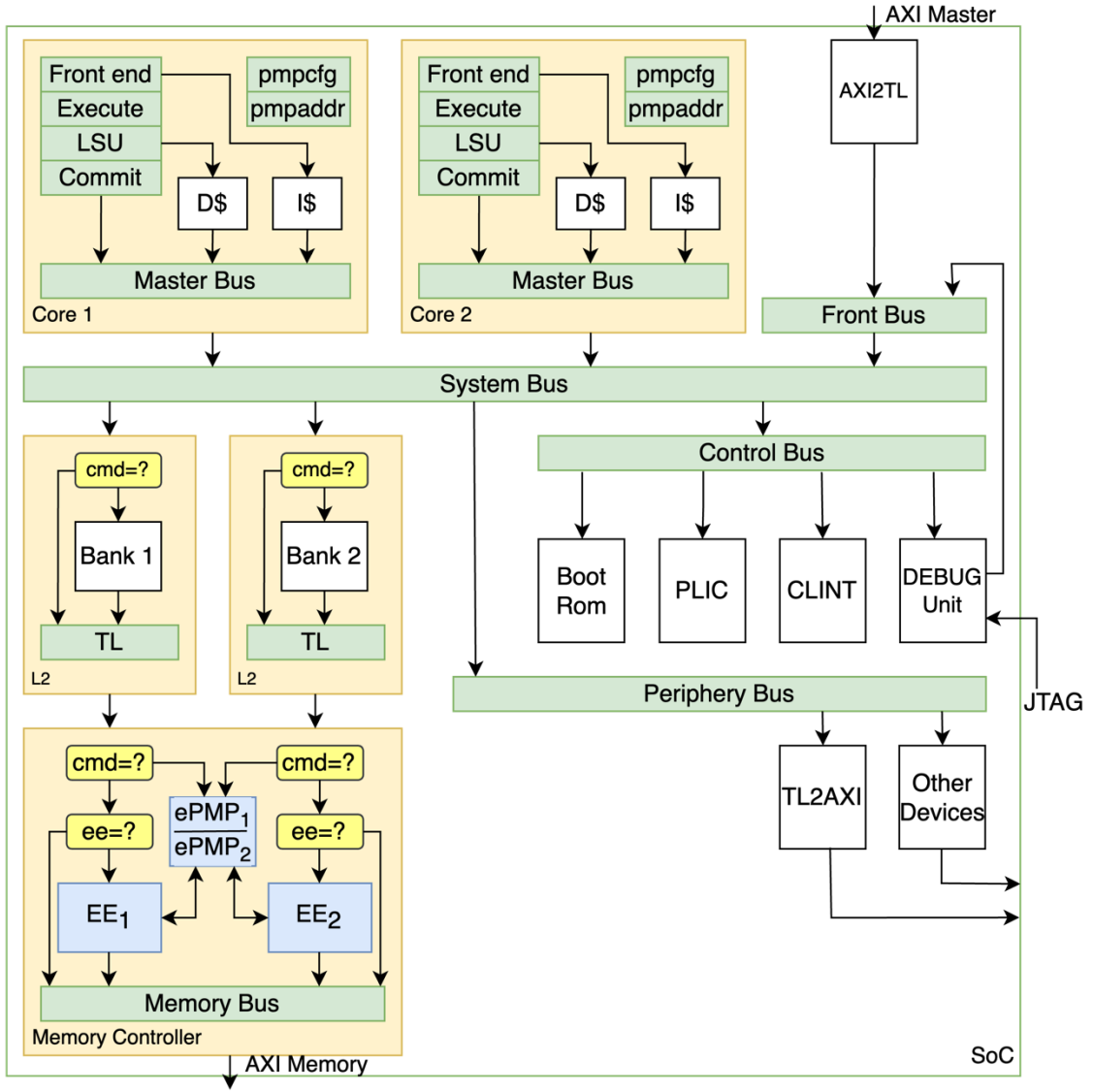
10 Add LUT to L2 Cache to either handle normal data request or forward request to Memory contorller

11 Finally, updated memory controller to update the ePMP

Extending RISC-V Keystone To Include Efficient Secure Memory

Secure Memory Example

- On Reset - ZSBL will load Keystone secure SM
- SM will configure pmpaddr and pmpcfg
- Modified Keystone that all enclave will set encryption bit
- On CSRW $pmpaddr_x$
  - the execute stage will buffer address
  - the commit stage request is sent to MC
- On Write - Load Store Unit(LSU) will invoke the PMPChecker
- Write is requested
- Assume write-through caches for this example
- Data is encrypted and integrity tree is updated
  - Up to this point data are **not** encrypted
- Encrypted data is sent to external memory
- SoC Boundary
- Caches



Continuing our Work

- Complete our FPGA implementation
- Efficiency
  - Area and Power
  - Utilizing existing RISC-V primitives
  - Critical path has minimal changes