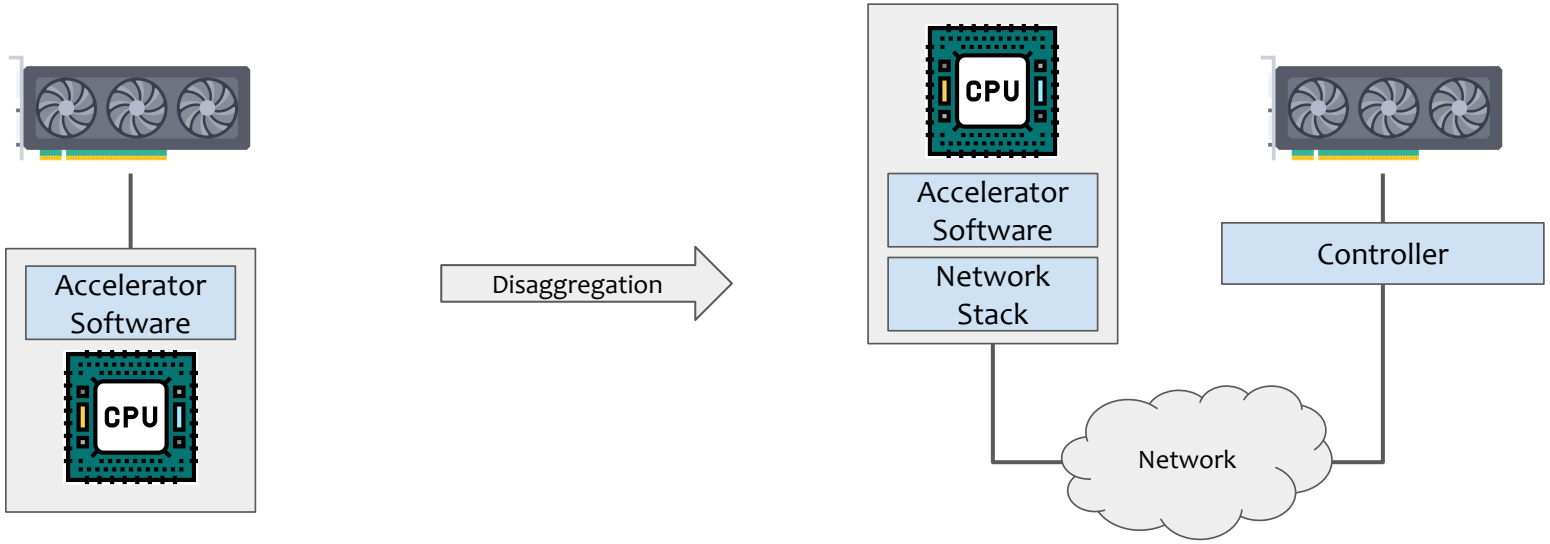
Protecting H/W and S/W Interactions for Network-Attached Accelerators

Maximilian Jäcklein
Advisor: Harshavardhan Unnibhavi
Supervisor: Prof. Pramod Bhatotia
Systems Research Group
https://dse.in.tum.de/

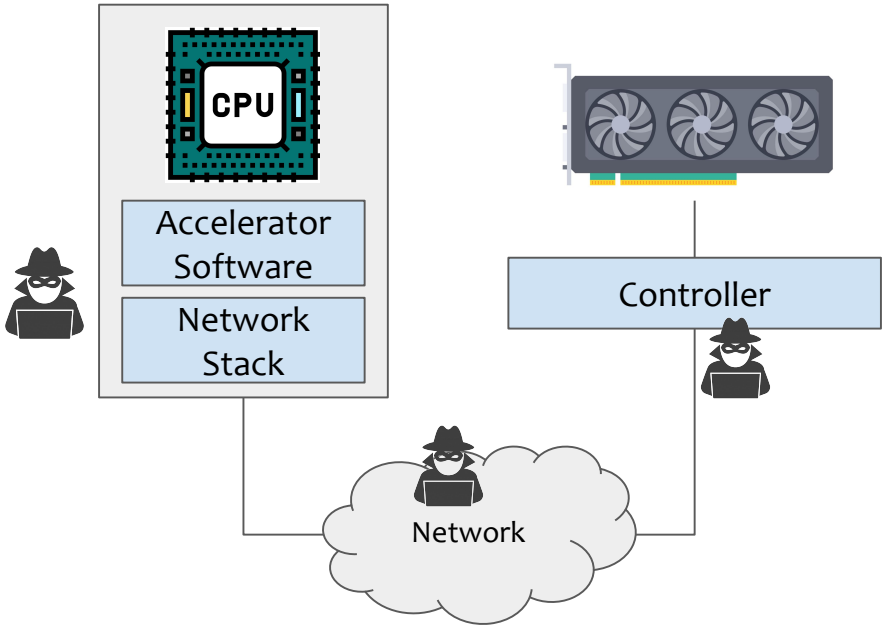
14.04.2025 - 14.08.2025

Context: Network-attached Accelerator



Lack of support for network-attached accelerator with existing protocols (e.g., PCIe)

Context: Security



Need for protection of application data and code during all stages of computation

Context

Network-Attached Accelerators

- Develop network stack for accelerator (e.g., Beehive [5])
- Change accelerator S/W and H/W
- No security measures

Confidential Accelerators

- Built on existing interconnect
  - For example: TDISP on top of PCIe (e.g., used in AMD SEV-IO [1])
- Rack-Scale
  - Connected via PCIe (e.g., HETEE [2])
- Changes to Accelerator Software
  - Accelerator API (e.g., Graviton [3])
  - Driver (e.g., SGX-FPGA [4])

Existing approaches do not allow access to confidential network-attached accelerators

[1] SEV-TIO: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/sev-tio-whitepaper.pdf
[2] Hetee [S&P 20]: https://heartever.github.io/files/hetee.pdf
[3] Graviton [OSDI 18]: https://www.usenix.org/system/files/osdi18-volos.pdf
[4] SGX-FPGA [DAC 21]: https://ieeexplore.ieee.org/document/9586207
[5] Beehive [Micro 24]: https://arxiv.org/abs/2403.14770v5

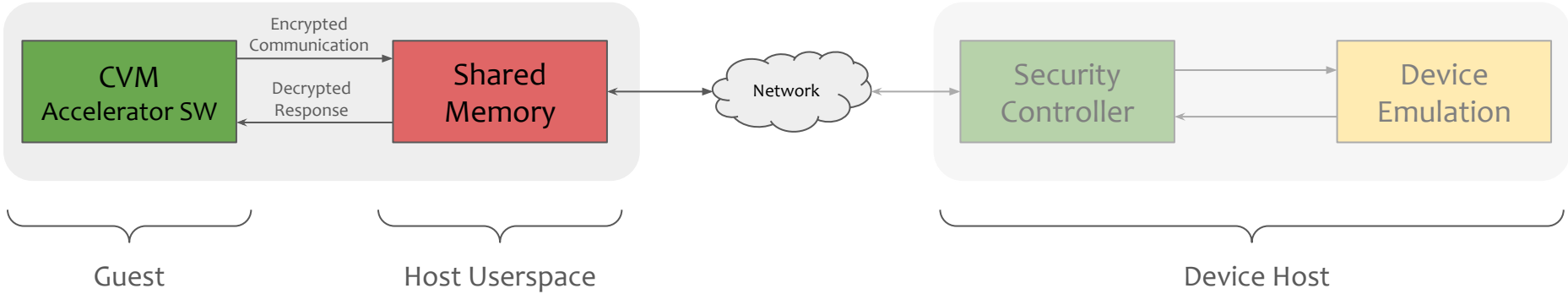
Problem statement

Provide trustworthy access to a network-attached accelerator without changes to accelerator SW and HW

Thesis' Scope

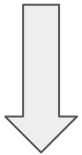
Enabling software support for accessing network attached accelerators transparently and securely

System Overview



Design Challenges

Transparency
System hides network communication from driver (Driver assumes CPU-attached)

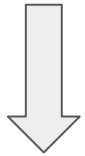


Intercept at communication interface

Security
Confidentiality, Integrity, and Freshness of communication

AES-GCM + message counter

Minimal Modifications
Existing device services (Kernel, UEFI) remain untouched

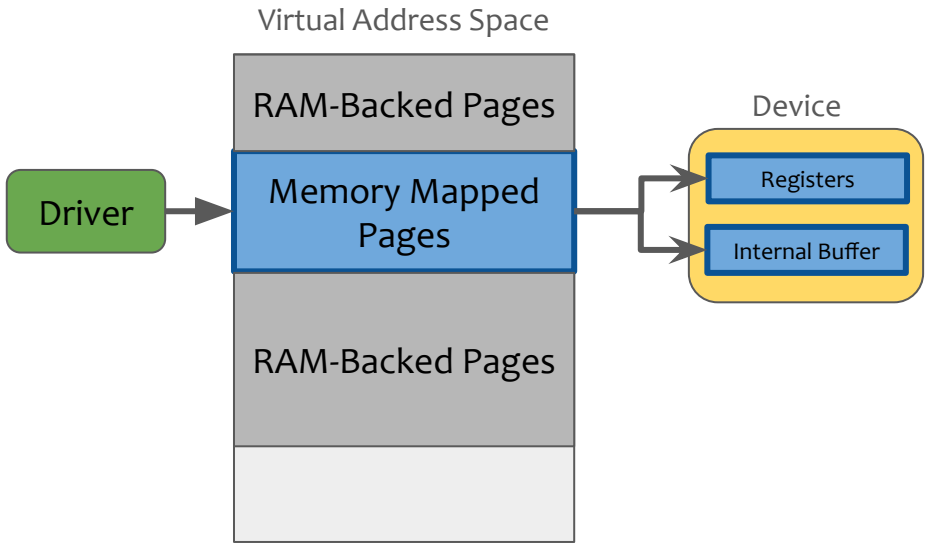


Pseudo device

Background: Device Accesses

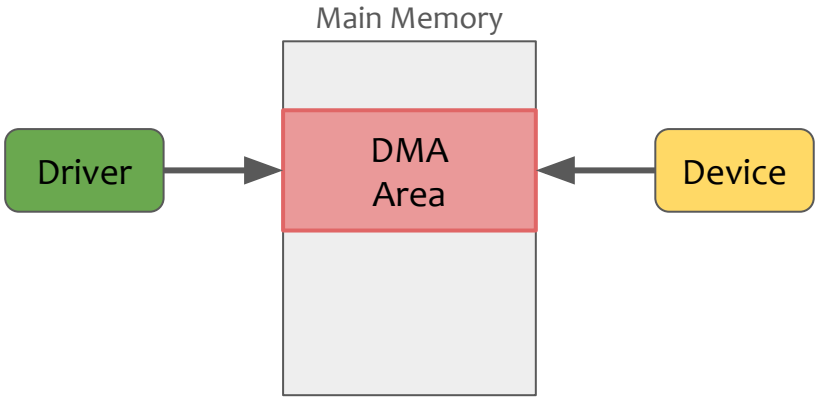
MMIO

Driver has direct access to Device Internals (register-sized)

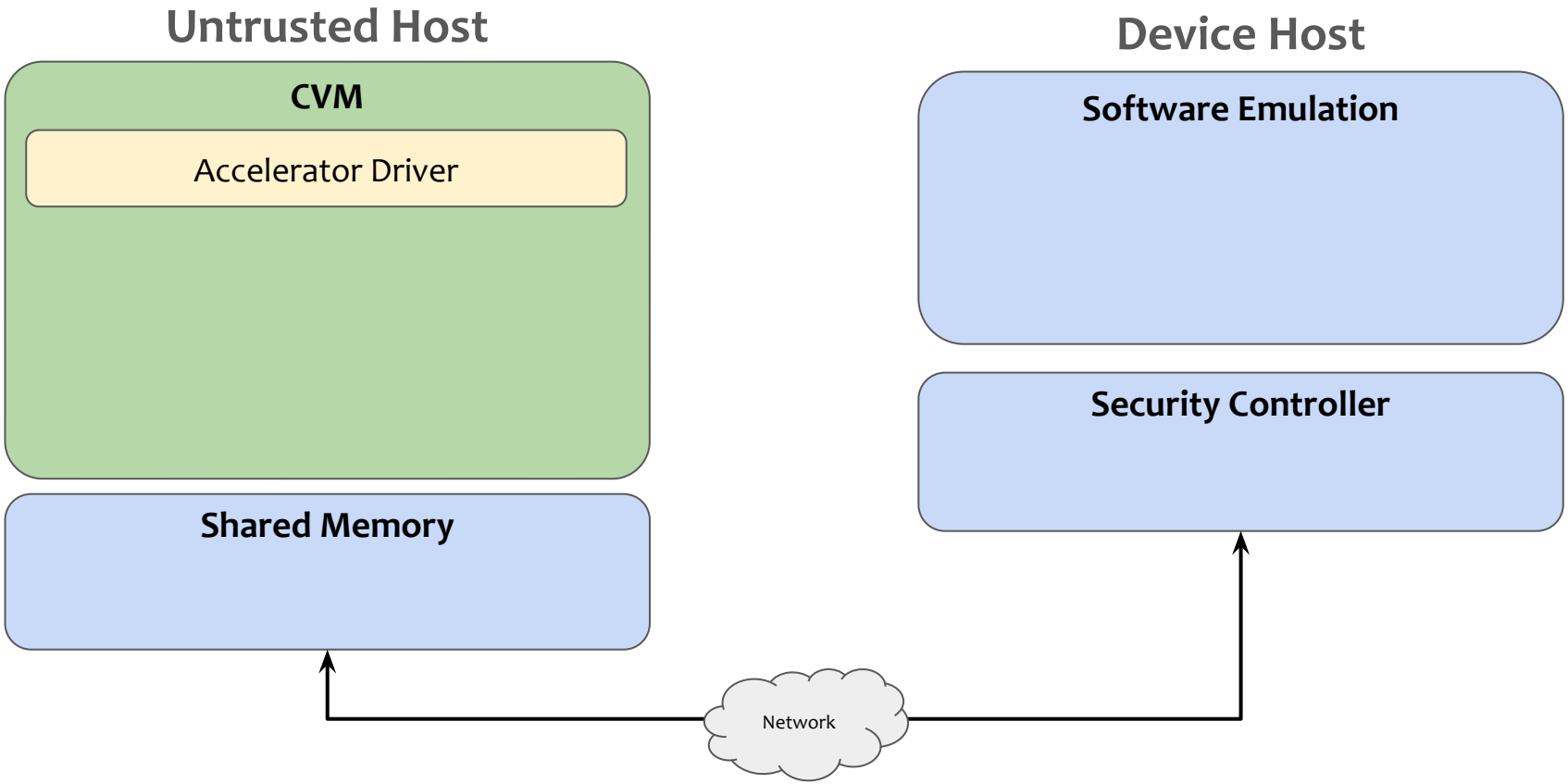


DMA

Driver and Device share Memory area (bulk-sized)

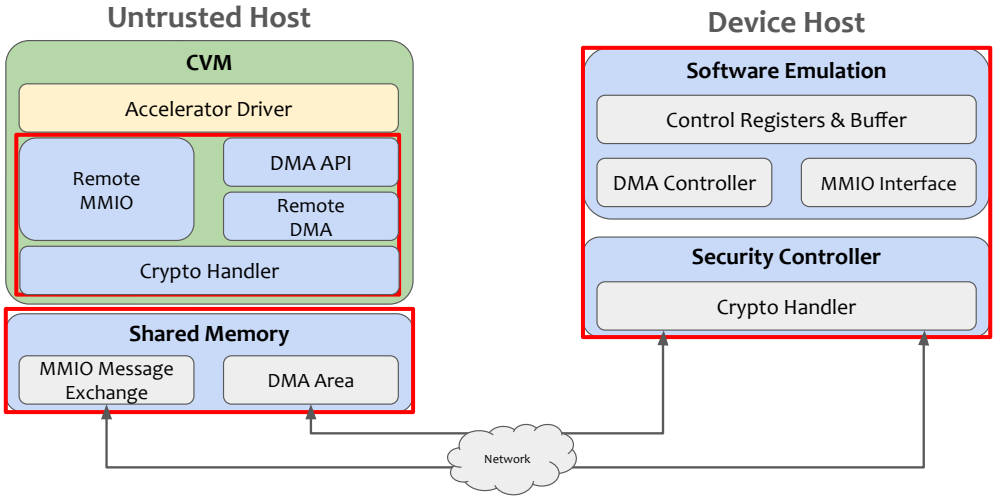


System Design

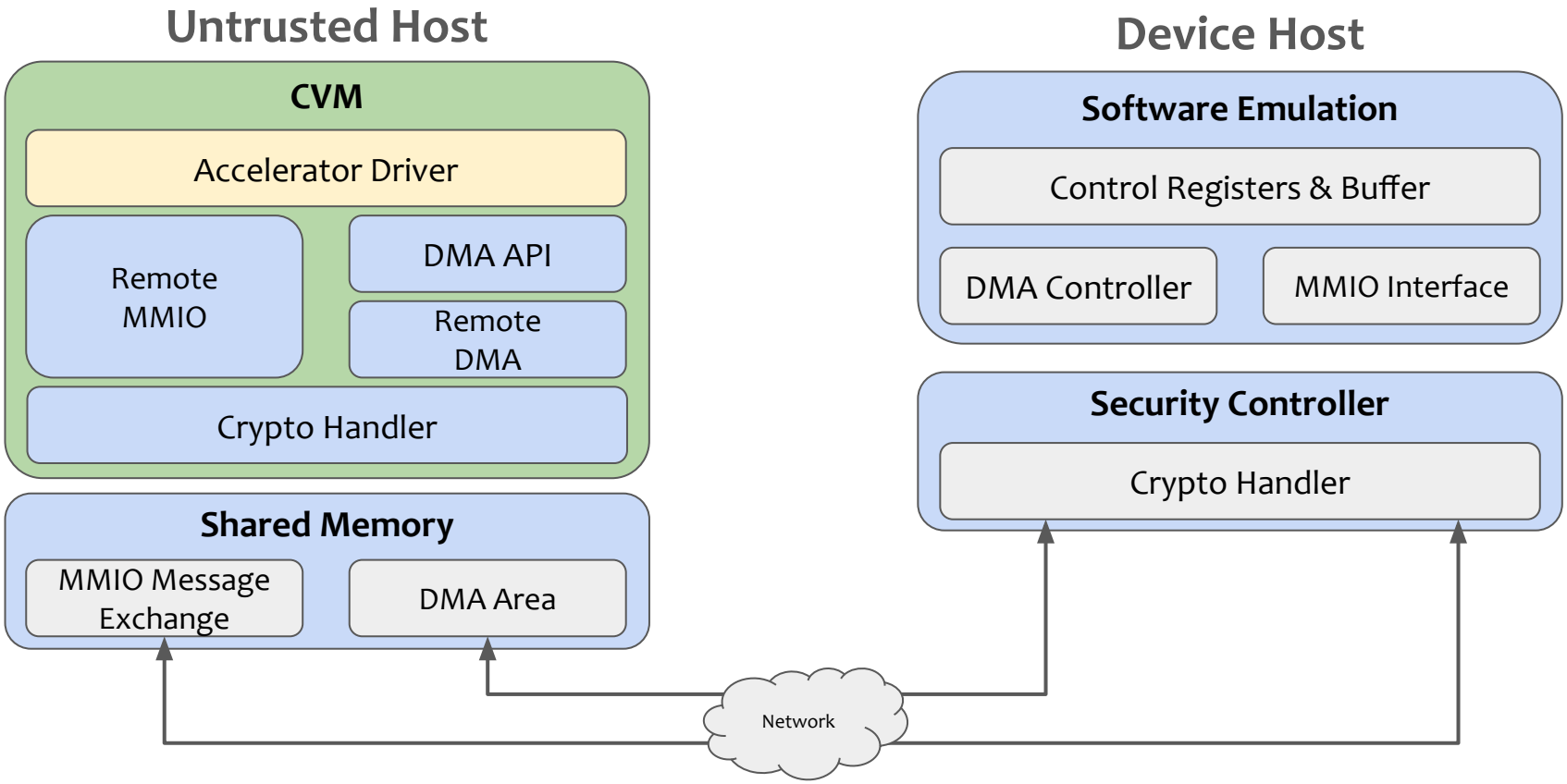


Design Components

- CVM Communication Software:
  - Intercepts MMIO/DMA attempts by the driver
  - Converts and encrypts information
  - Returns device's responses
- Shared Memory:
  - Bridges gap from unprotected host to CVM
  - Networking capabilities
- Device-side Software Stack:
  - Endpoint of network connection
  - Decrypts requests or encrypts device response
  - Device software emulation



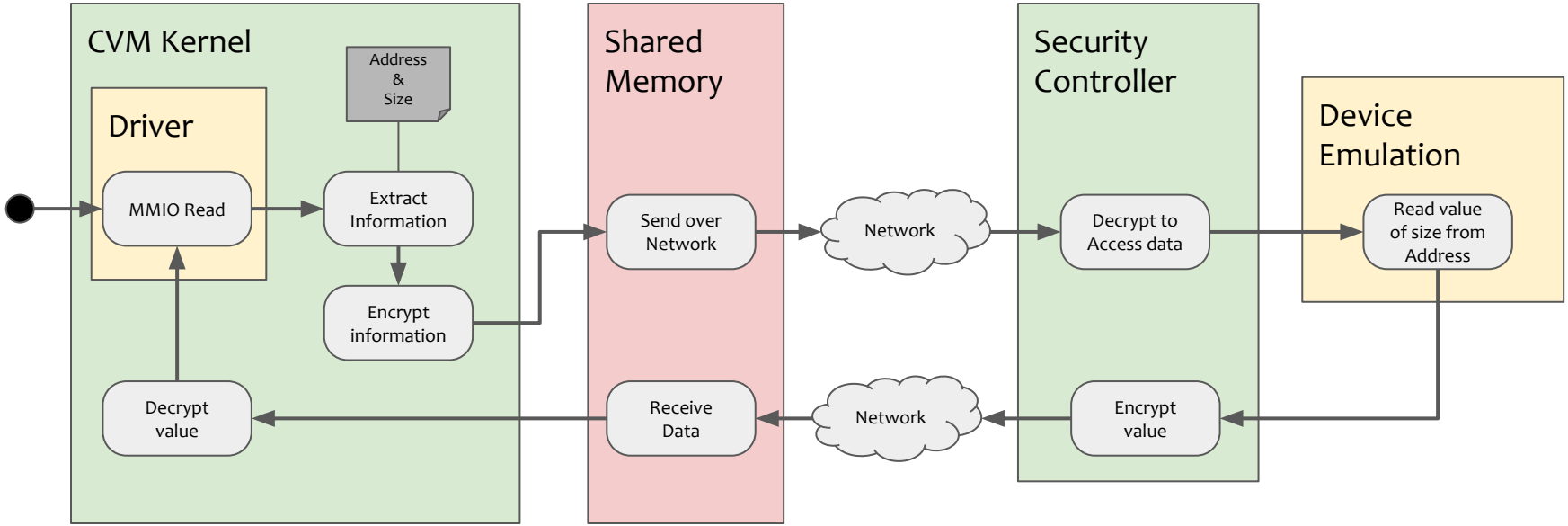
System Design



Modified / Addition

Unmodified

Example Workflow: MMIO Read



Evaluation: Questions

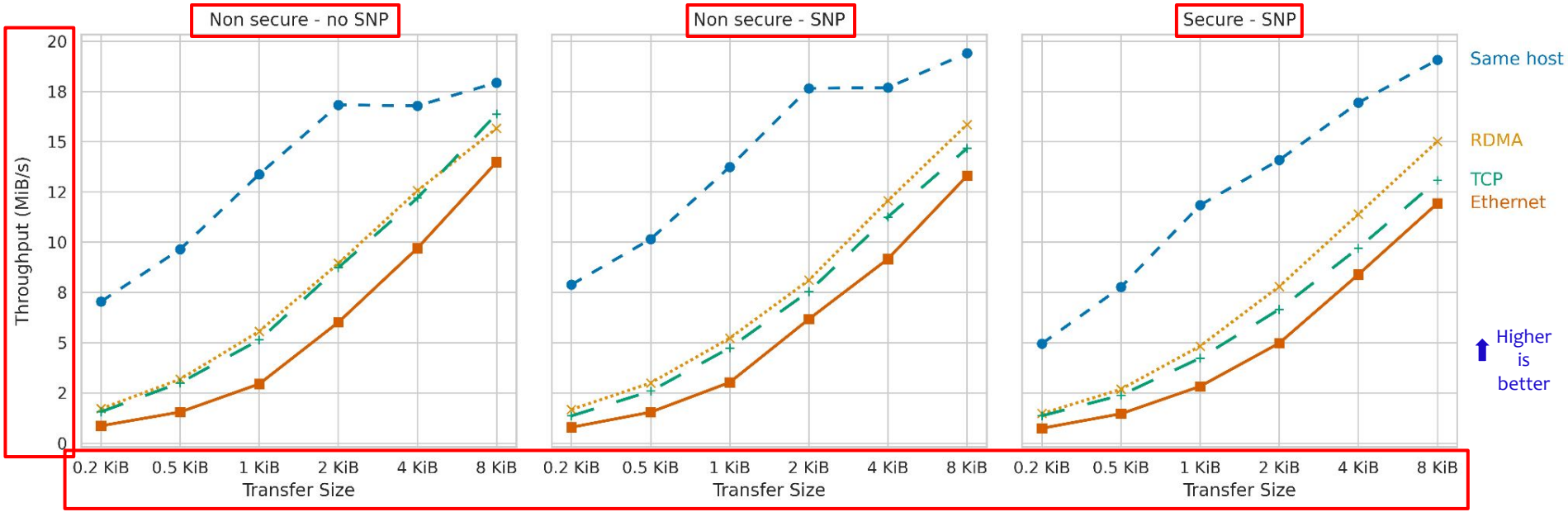
1. **End-to-end overhead?**
2. **Register-size access overhead (MMIO)?**
3. Bulk-data transfer overhead (DMA)?
4. Security (encryption and authentication) overhead?
5. Transportation types differences?

Evaluation: Experimental Setup

- CVM host:
  - AMD EPYC 7413 (24 cores, 1.5 GHz)
  - SEV-SNP hardware extensions
  - Crypto: Kernel API with AES-NI
  - Hypervisor: QEMU with 8 vCPUs and 8 GB RAM
- Device emulation host:
  - AMD EPYC 7713P (64 cores, 1.5 GHz)
  - Crypto: OpenSSL with AES-NI
- Network Connection:
  - Direct connected via two E810-C (support RoCE)
- Baseline:
  - Same host for CVM and Device (EPYC 7413)

DMA Throughput

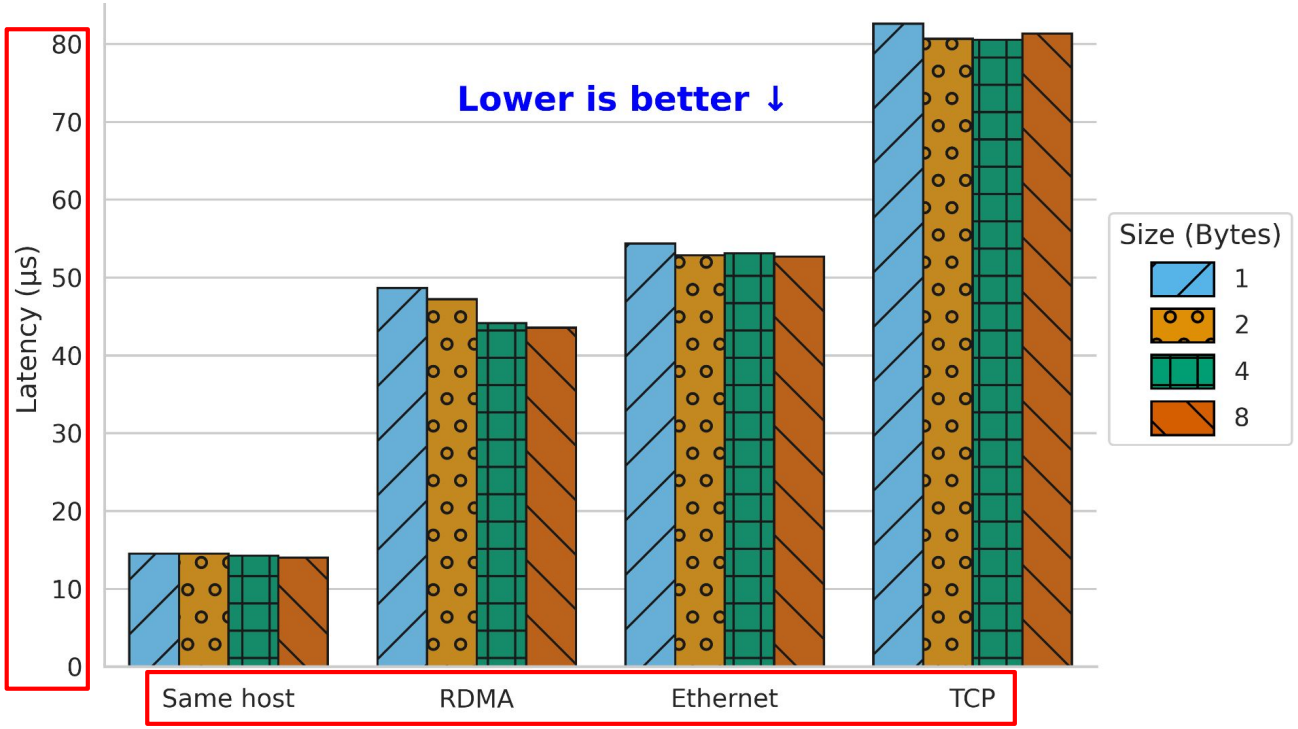
DMA transfer (both directions) throughput across security levels and sizes



=> Secure DMA accesses introduce low overheads but are affected by the underlying software network stack

MMIO Latency

MMIO read access latency across different connection types and sizes



=> Secure MMIO accesses introduce low overheads but are affected by the underlying software network stack

Conclusion

Existing approaches do not provide **secure** access to **network-attached** accelerators

This thesis contributes:

- Software implementation
  - Changes to communication layer (MMIO and DMA) of kernel
  - User space application for data exchange and networking (Shared Memory)
- Multiple network transportation types
- Device emulation

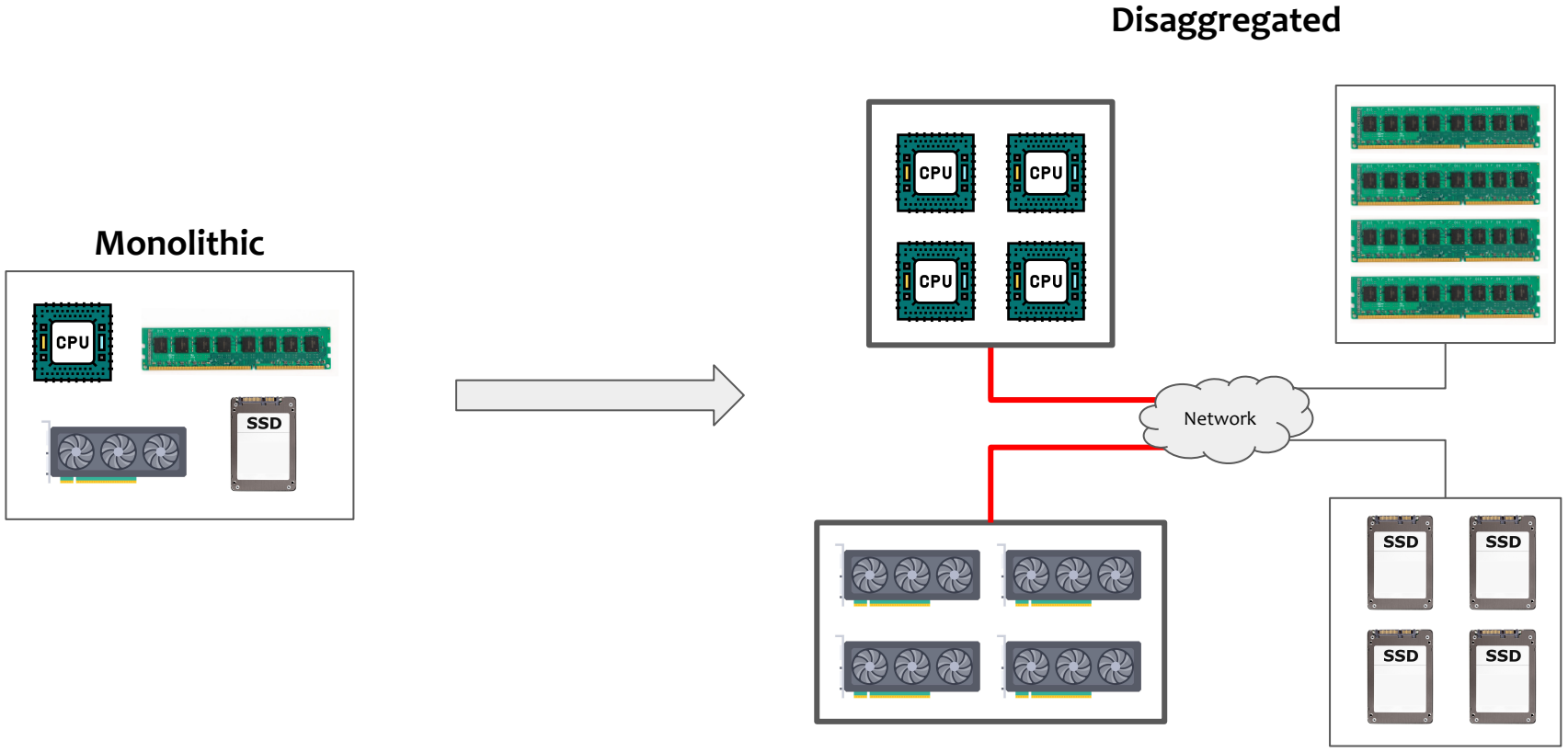
Future work:

- Interrupt support
- Network overhead mitigation

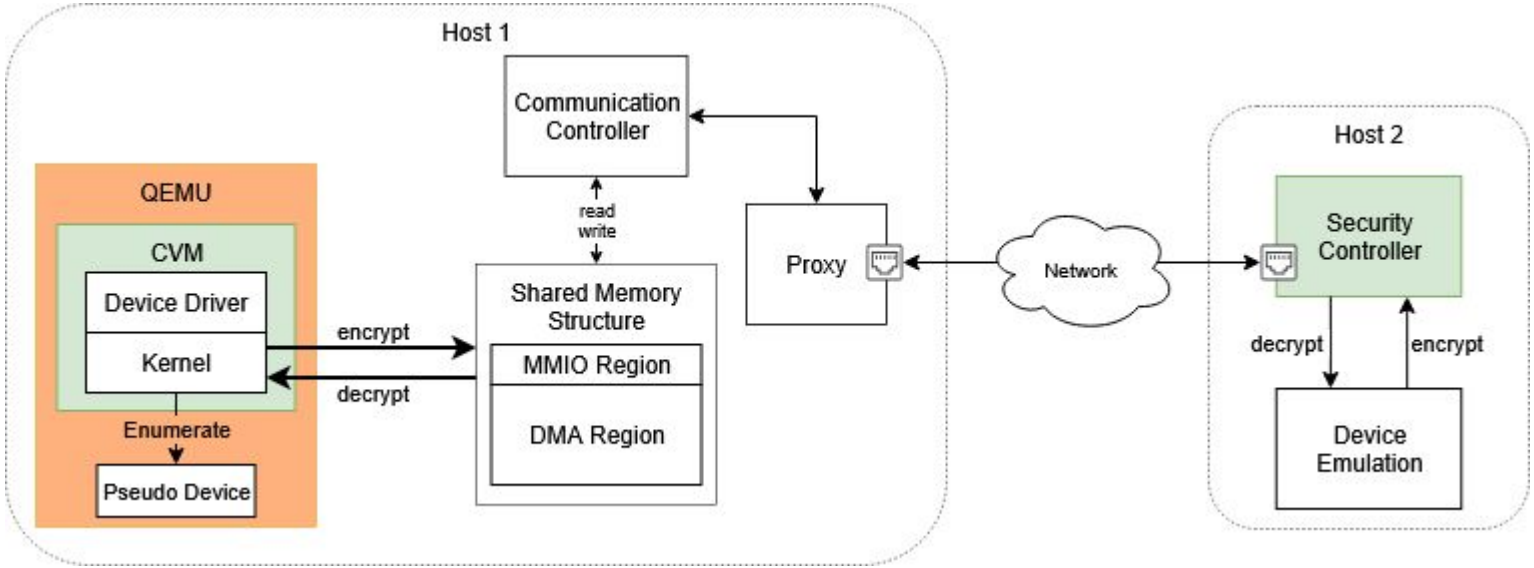
Source code:
https://github.com/harshanavkis/jigsaw-overall/pull/2

Backup

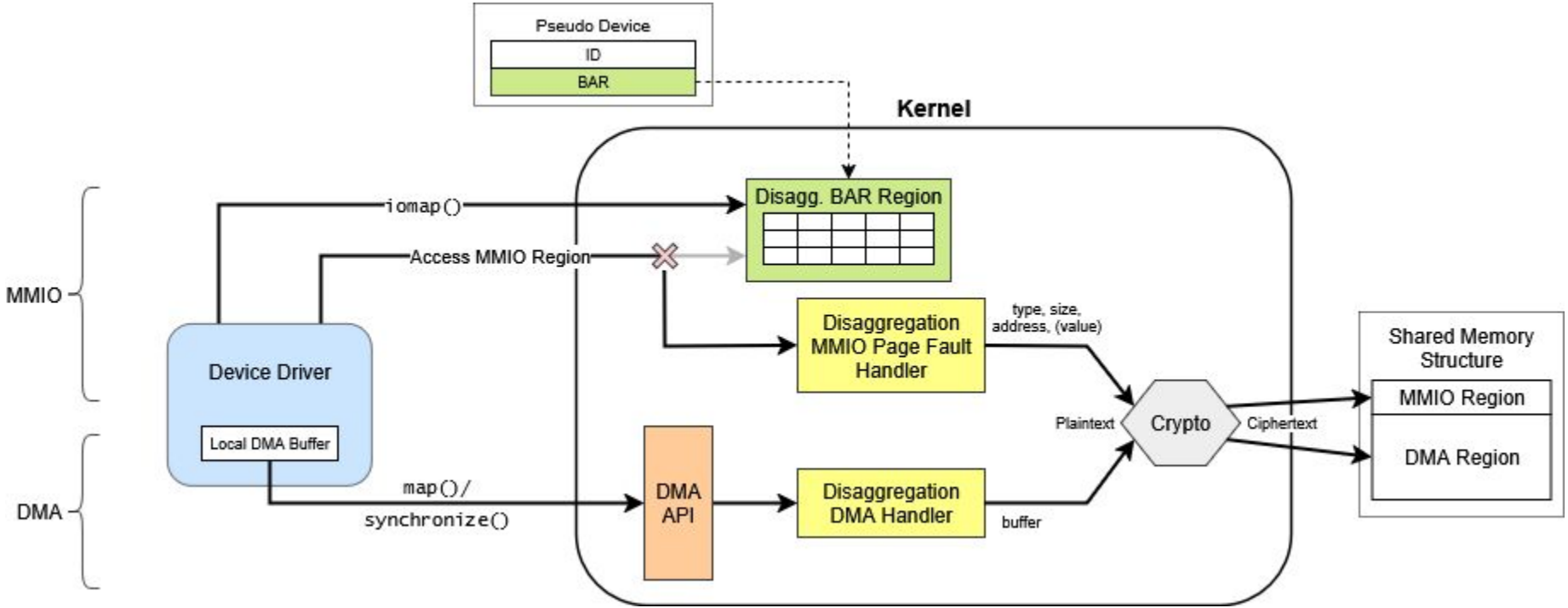
Context: Disaggregation



System Design



Implementation



MMIO Read Latency