ccAI: A Compatible and Confidential System for AI Computing

Chenxu Wang[12], Danqing Tang[3], Changxu Ci[3], Junjie Huang[1], Yankai Xu[1],
Fengwei Zhang[1]✉, Jiannong Cao[2], Jie song[3], Shoumeng Yan[3]✉, Tao Wei[3], Zhengyu He[3]

1Southern University of Science and Technology, 2The Hong Kong Polytechnic University, 3Ant Group

AI Computing is Popular

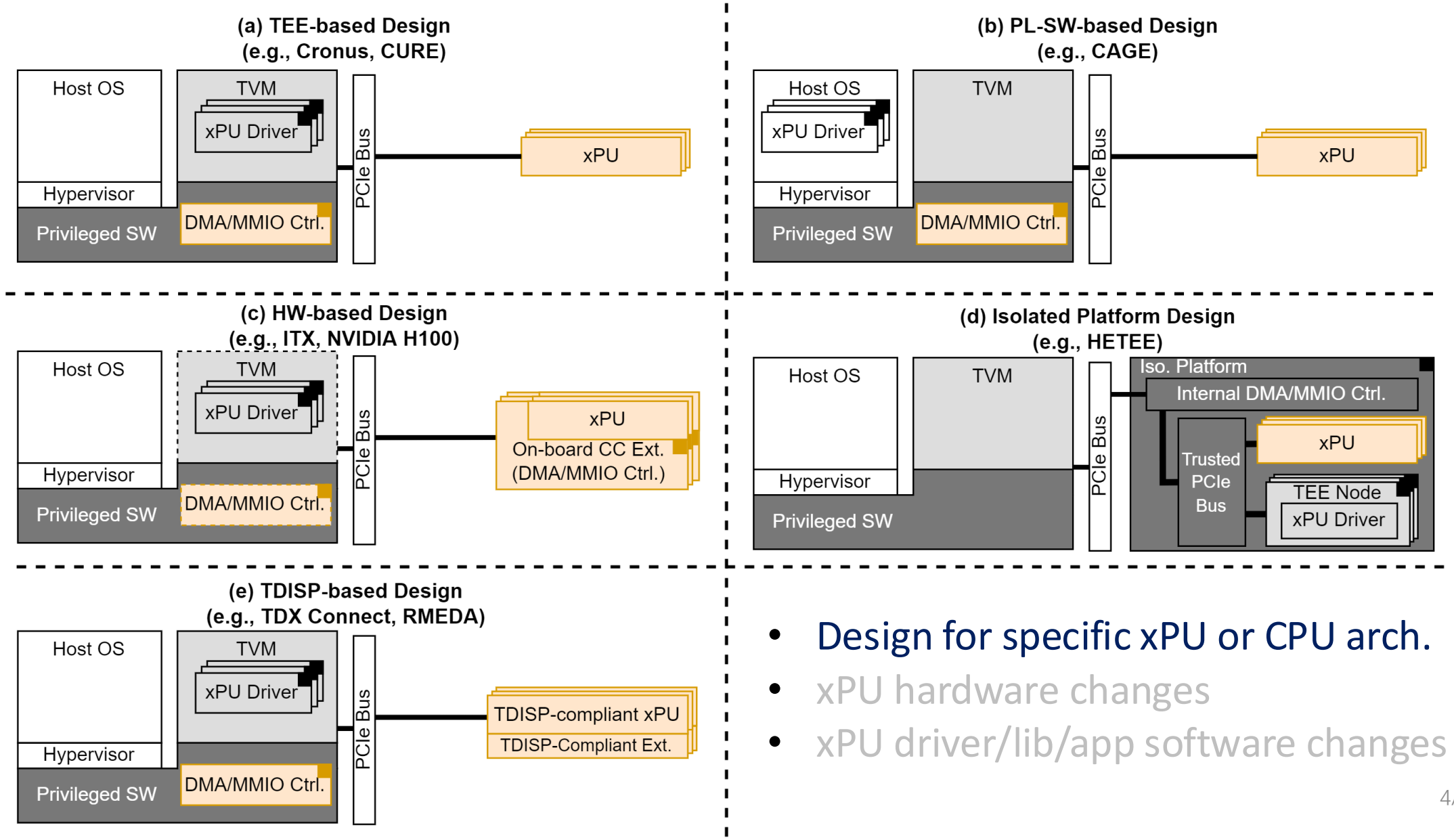
- Wide application scenarios
  - Large Language Models (LLMs): ChatGPT, Deepseek
  - Image & Video Processing: Sora2
- Serves different heterogeneous clouds
  - Google Cloud, Micorsoft Azure, Alibaba Cloud…
- xPU & PCIe: **Key components and bridge** for AI acceleration
  - GPU, NPU, TPU, FPGA-based CNN/DNN Accelerator
  - Peripheral Component Interconnect express (PCIe)

xPU-based AI Computing is Vulnerable

- xPU environment is easy to be compromised
  - General xPU lacks confidential computing support
  - xPU driver and library can be buggy
  - **Problem**: Cloud users cannot trust xPU environment

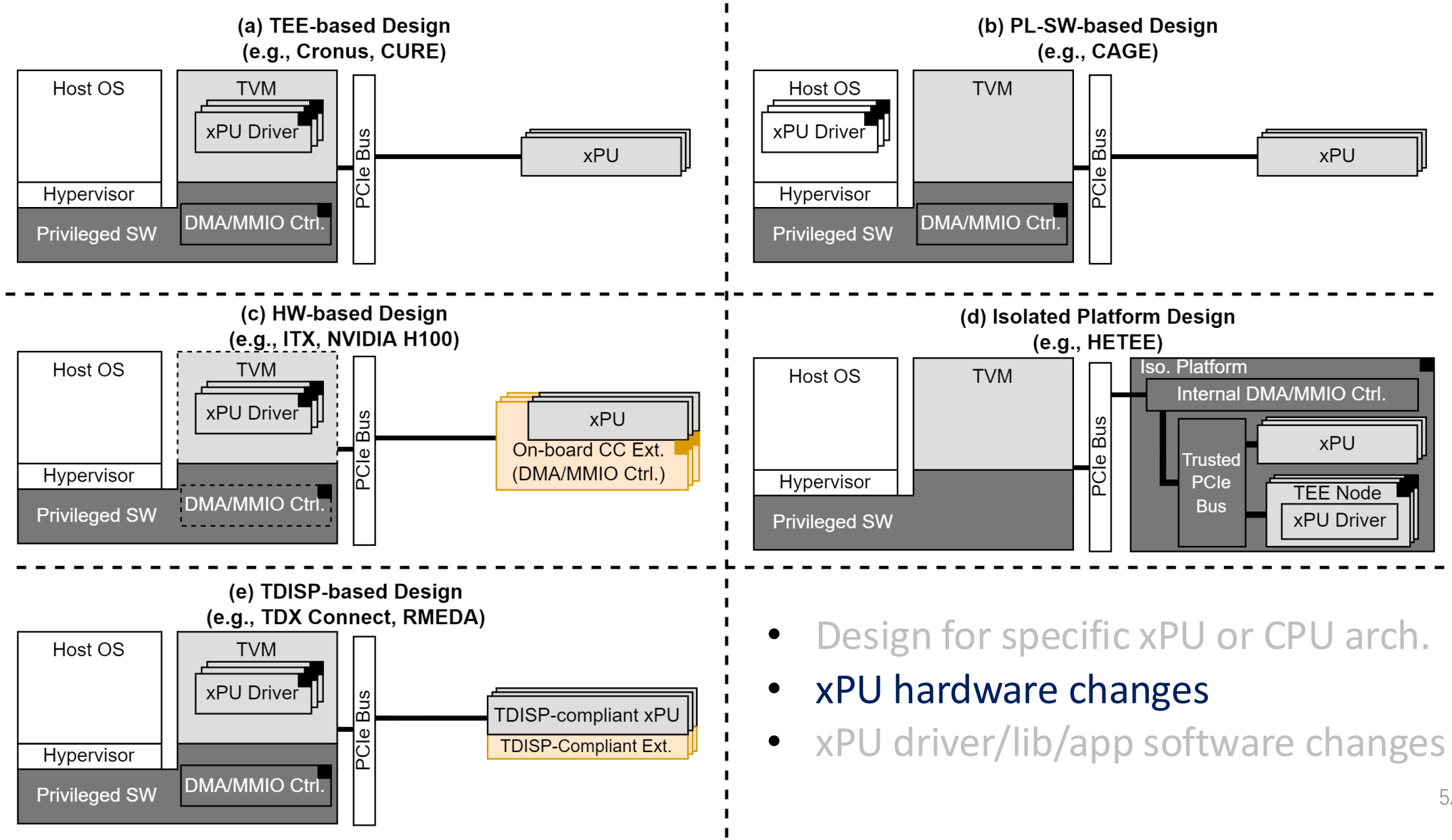
- A critical solution: xPU Trusted Execution Environment (xPU TEE)
  - NVIDIA Hopper GPUs (H100): First commercial xPU TEE
  - xPU data/model **Confidentiality** and **Integrity**
  - xPU computing with **Isolation**
  - xPU-equipped system with **Authenticity**

Motivation: xPU TEEs Face Compatibility Problem



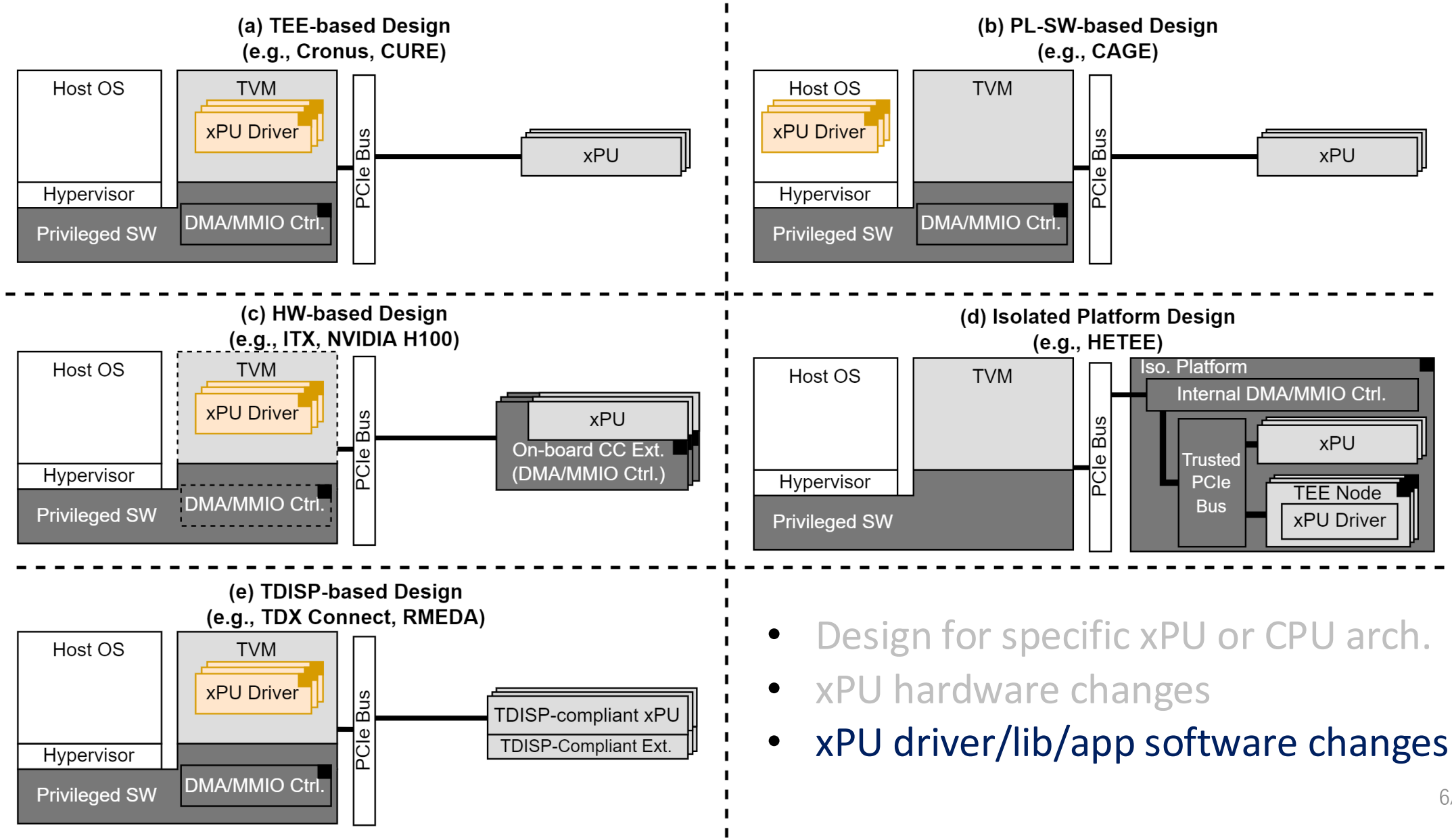
- Design for specific xPU or CPU arch.
- xPU hardware changes
- xPU driver/lib/app software changes

Motivation: xPU TEEs Face Compatibility Problem



- Design for specific xPU or CPU arch.
- xPU hardware changes
- xPU driver/lib/app software changes

Motivation: xPU TEEs Face Compatibility Problem

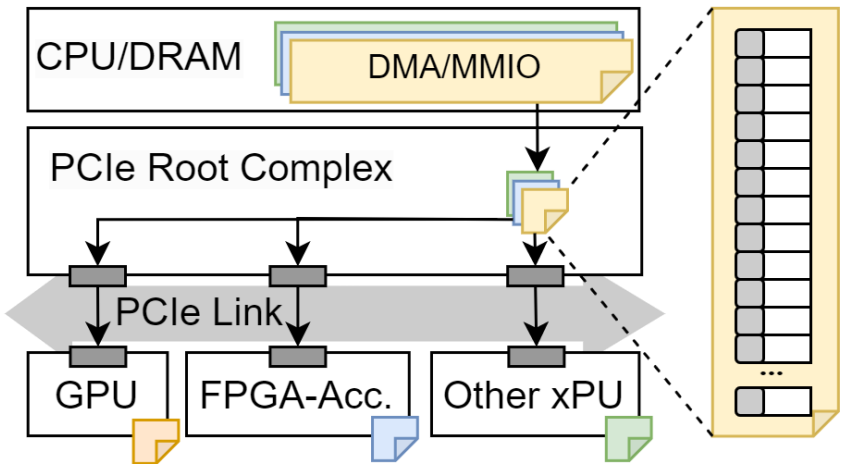


- Design for specific xPU or CPU arch.
- xPU hardware changes
- xPU driver/lib/app software changes

Motivation to Primary Goal: High Compatibility

- Problem: How to design a compatible xPU TEE framework?
  - Specific xPU/CPU arch & xPU HW changes → Support **multi-type** xPUs
    - Different xPU may support **unique** xPU software stacks
    - xPU software stacks **lack confidentiality guarantees**
  - xPU driver/lib/app changes → Ensure **user transparency**
    - **No changes** for reducing developer's engineering effort

- Solution:
  - Multi-type: Design protection on PCIe channel, focusing on PCIe packet
    - A bottom-layer unit for DMA/MMIO
    - Commonly used in varied xPU/CPU
  - User transparency:
    - A "middleman" in CPU-side TVM



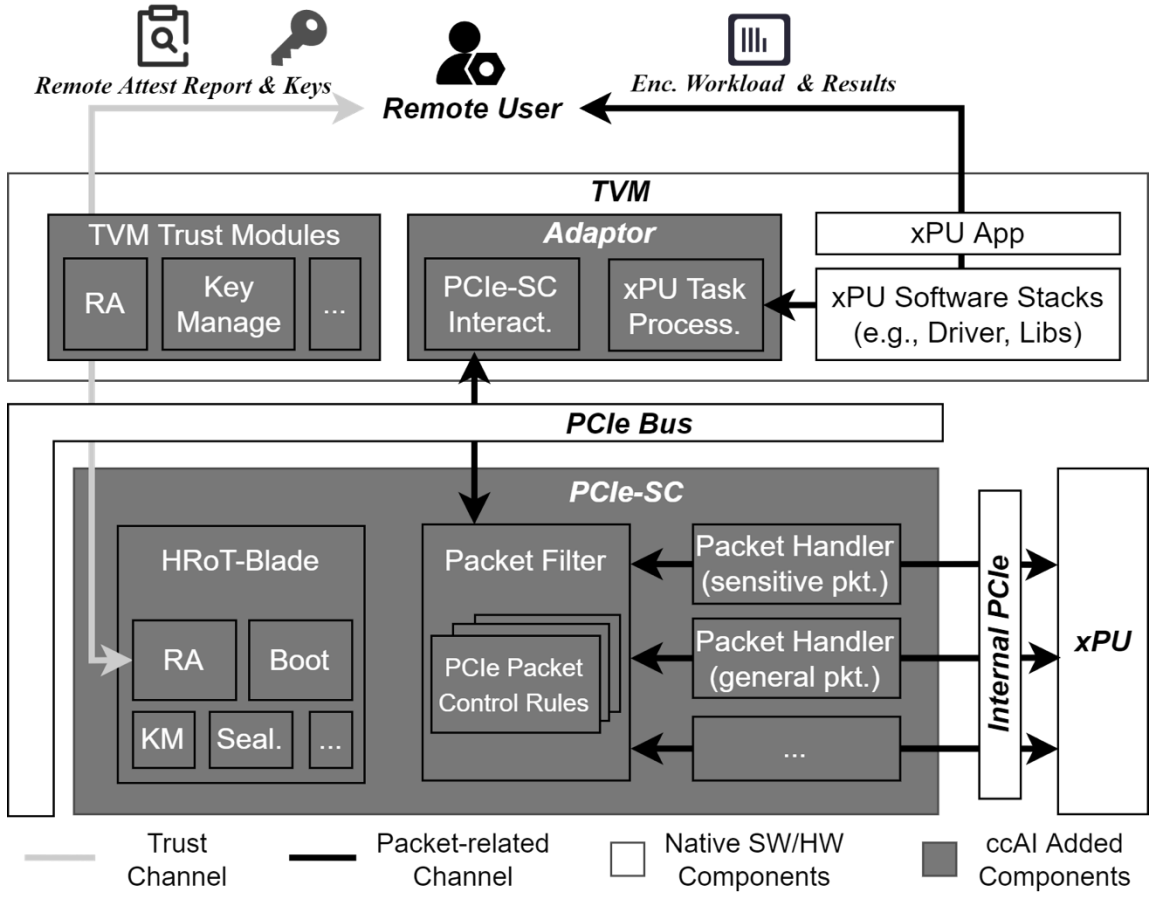
ccAI: High Compatibility Solution for xPU TEE

ccAI Design:

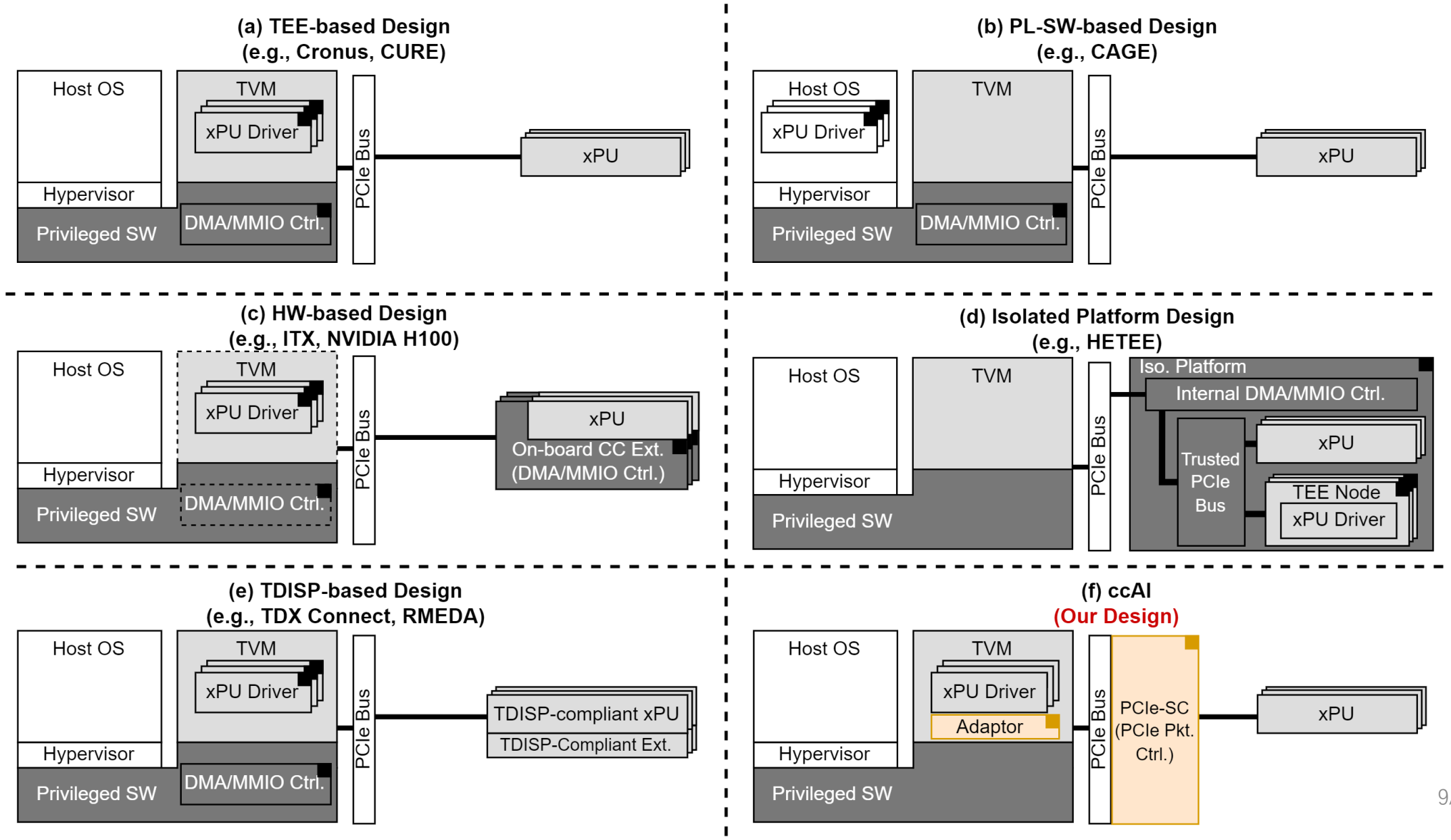
- **PCIe Security Controller (PCIe-SC)**
  - PCIe Switch, adapting varied xPUs

  → Confidential support
    - Attest, Enc, access ctrl, etc.
- **TVM-side Adaptor**
  - Adapting xPU SW with transparency

  → A "middleman" software for
    - Interacting with PCIe-SC
    - Securely initializing xPU application



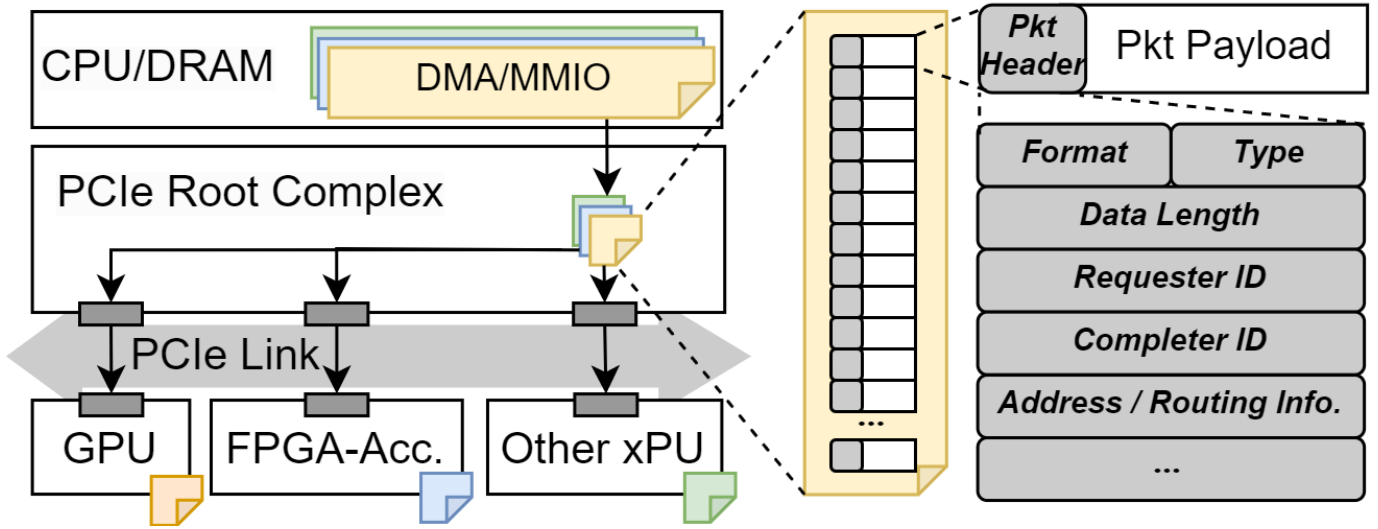
Design Comparison in Compatibility



Goal 2: Strong Security in xPU Computing

- Goal: Ensure Confidentiality/Integrity/Authenticity for xPU computing
  (basic TEE requirements)

- Problem: How to filter and manage PCIe packets?
  - Cannot design a one-size-fits-all solution, because PCIe packets are **complex**
    - Packets have different types and carry **diverse attributes (FMT, ID, Type, …)**
    - Also, same-type packets with **different attribute values** can be differently handled



Solution to Goal 2

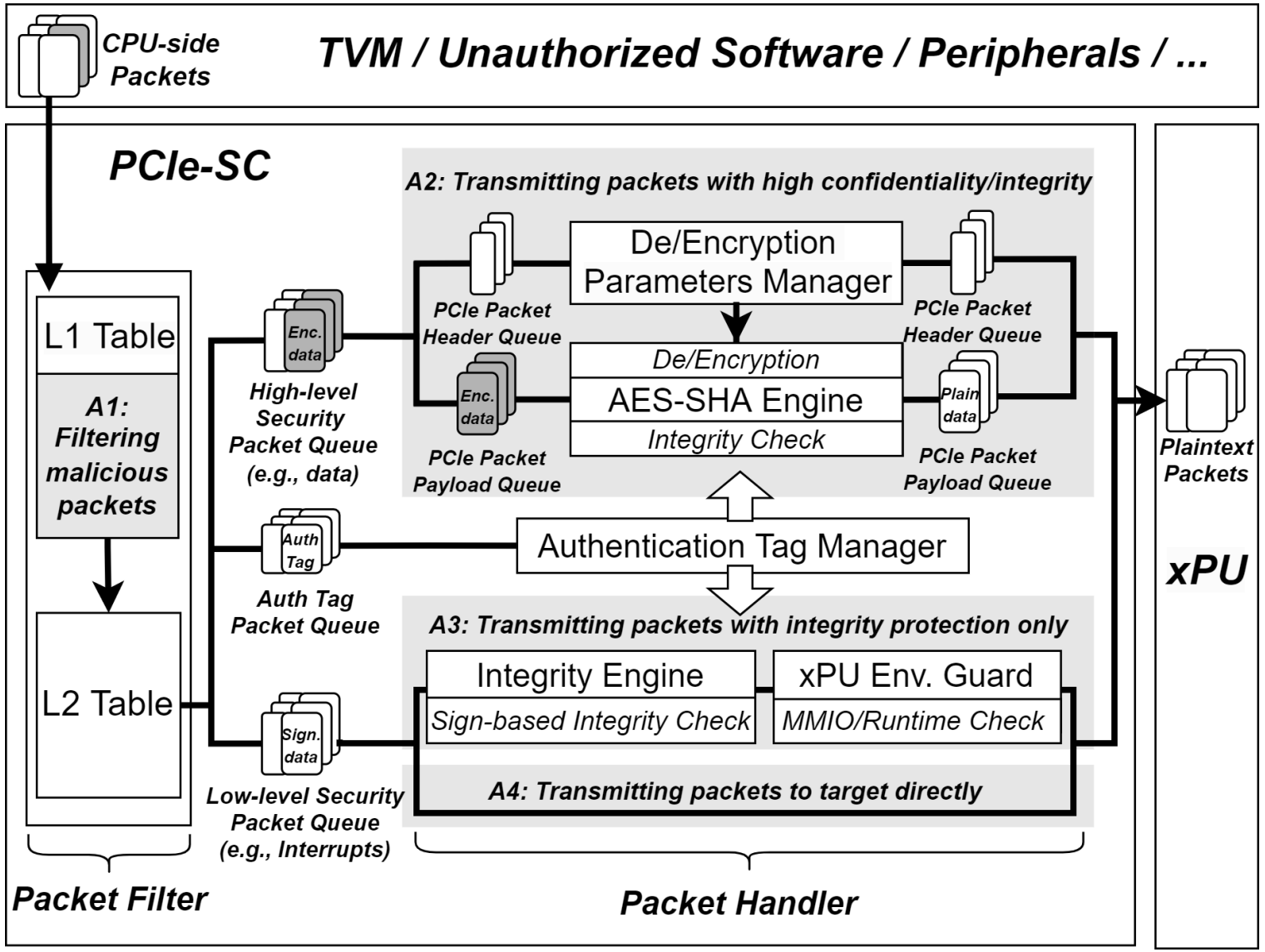
- Firstly, systematically analyze PCIe packets and propose security **categorization**
- These PCIe packets are categorized into four types, with corresponding actions:

Packet Access Permission	Actions
Prohibited	(*A1*) Disallow
Write-Read Protected	(*A2*) Integrity Check (Crypt.) + En/Decryption
Write Protected	(*A3*) Integrity Check (Plain) + Security Verify
Full Accessible	(*A4*) Transparent Transmission

- Processing packets with two major components
  - **Packet Filter**: Blocking malicious packets (A1) and classify authorized ones
  - **Packet Handlers**: Providing different security operations (A2-A4)

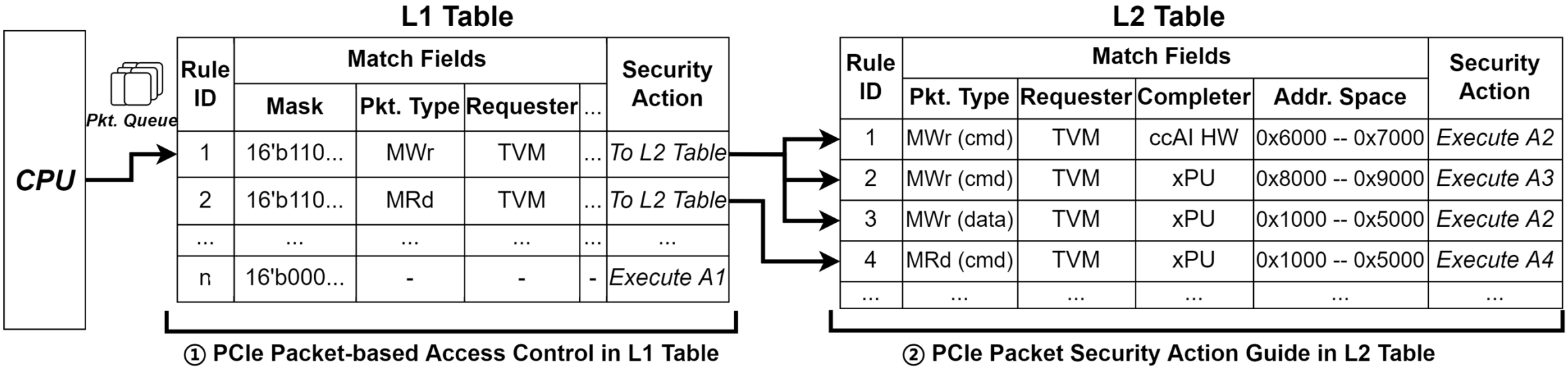
Solution in Goal 2

- Two major components
  - **Packet Filter**
    - L1 Table: Block
    - L2 Table: Classify
  - **Packet Handlers**
    - High security
    - Integrity check only
    - Direct transmission



Solution in Goal 2: Packet Filter

- L1 Table
  - Roughly check attributes, mainly **identify malicious packets**
- L2 Table
  - Check detailed attributes and values (type, ID, addr_space, etc.) for actions



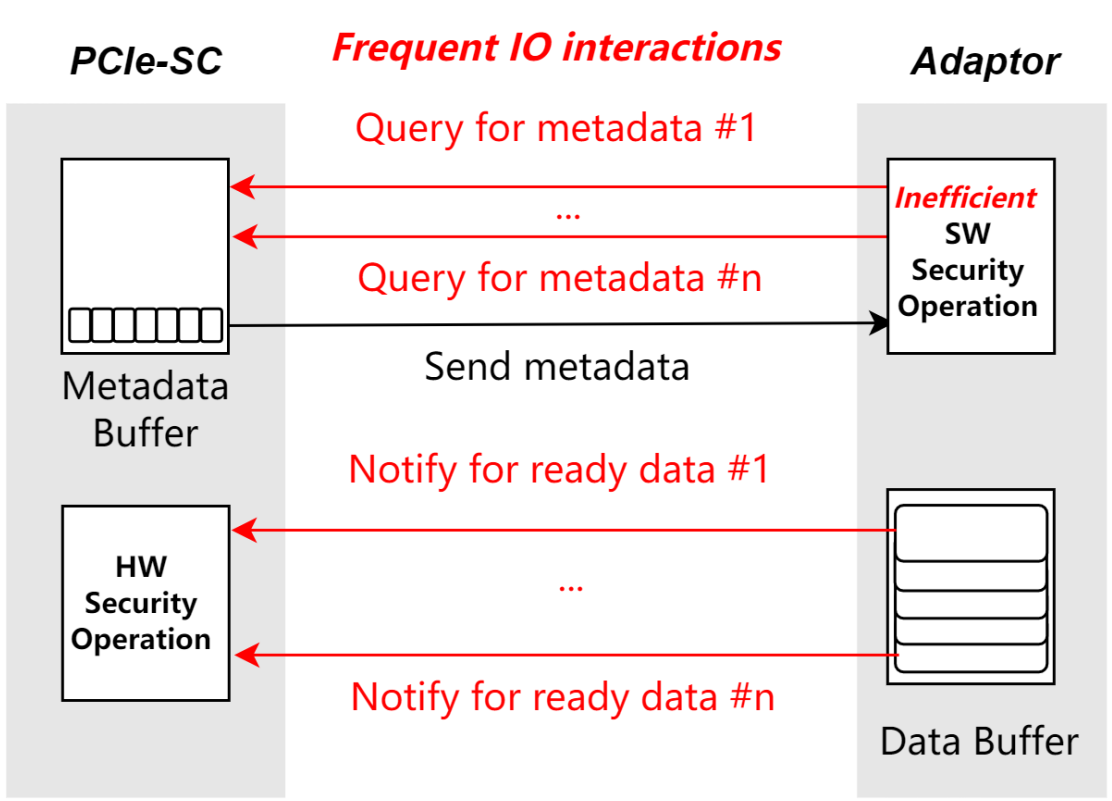
Solution in Goal 2: Packet Handlers

- Key observation: For processing different packets, the workflow is standardized
  - **Analyze** packet headers and authentication tags
  - **Extract** packet payloads and process
  - **Merge** header and processed payload together

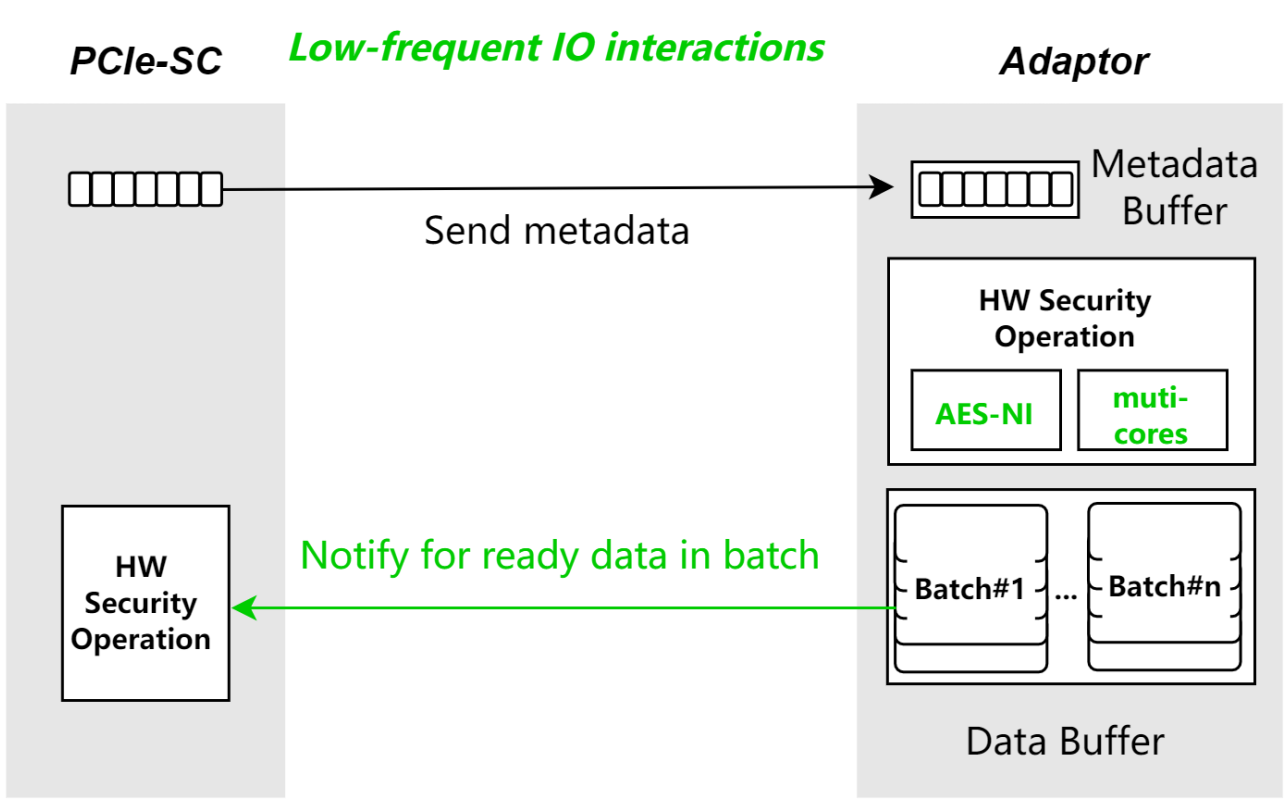
- Our handlers design
  - Control panels:
    - De/Encryption Parameters
    - Authentication Tags
  - Security operations:
    - AES/SHA engines ======> Will add more algorithms support
    - Environment guard: check MMIO status, reset env.

Goal 3: Performance Optimization

- Processing **I/O read and write** packets in batch
- **AES-NI**, and **multi-core allocation** for optimizing security operations



Non-optimized



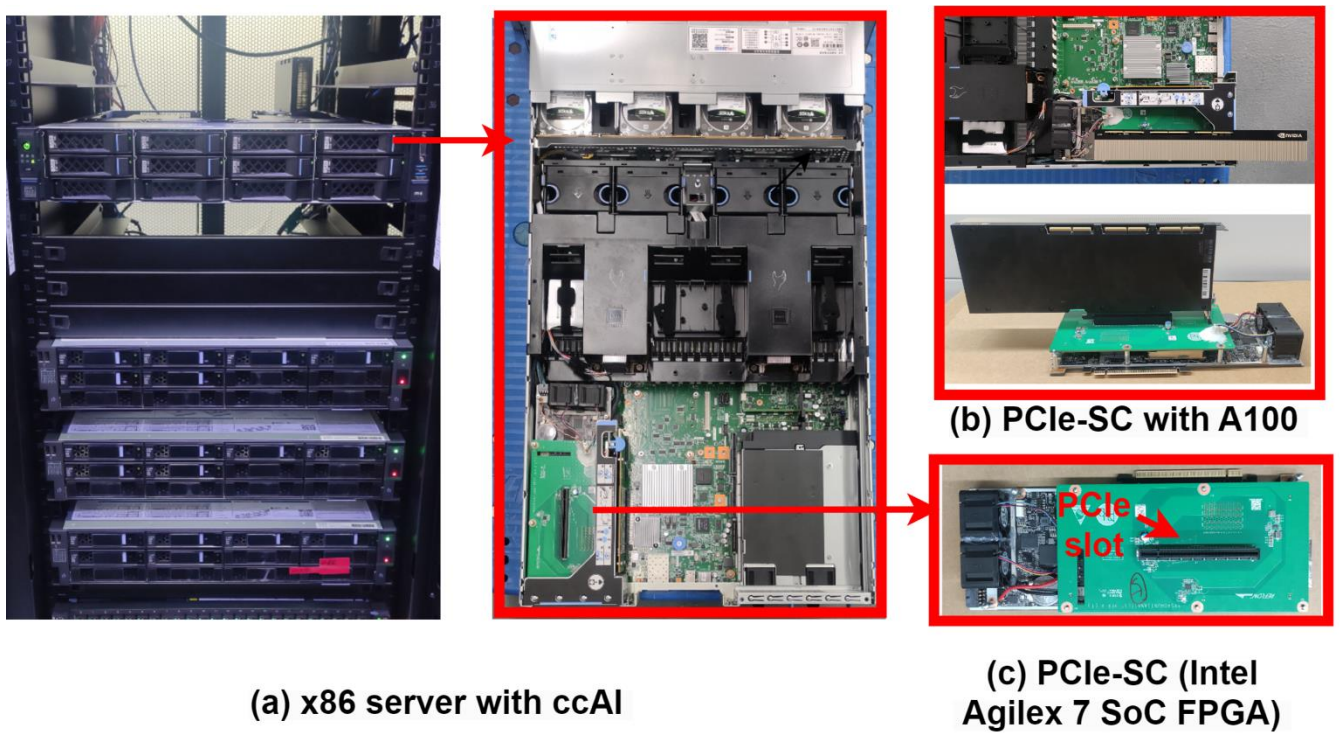
Optimized

Prototype Implementation

- Environment:
  - TVM: Intel Server (256GB Memory, 96 Cores) ====> Will support others
  - PCIe-SC: Intel Agilex 7 FPGA

- Trust Establishment
  - Self designed HRoT-Blade
    - Secure boot, attestation
  - Key management
    - SPDM keys, AES keys
  - Sealing in a chassis
    - Sensors =(I^2C)=>HRoT-Blade

Check our paper for details



(a) x86 server with ccAI

(b) PCIe-SC with A100

(c) PCIe-SC (Intel Agilex 7 SoC FPGA)

Security Evaluation

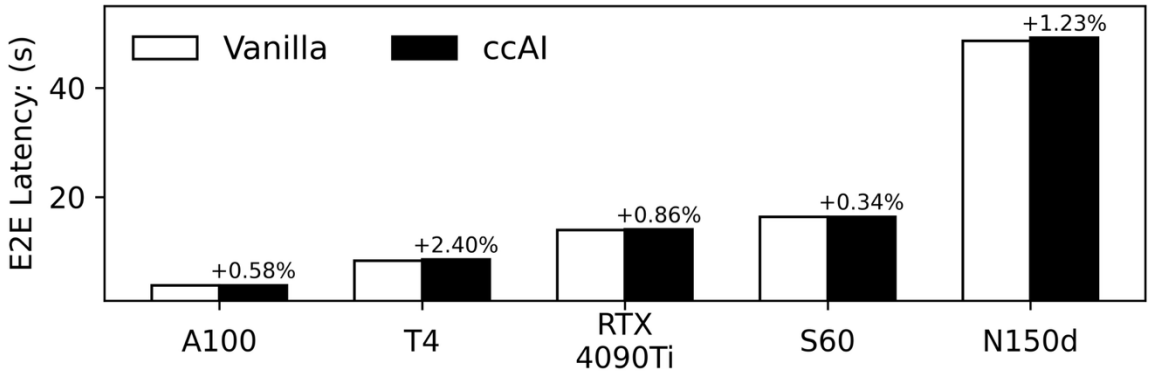
- ccAI defend against:
  - Access from host and unauthorized TVMs
  - Access from malicious devices
  - Physical attacks on PCIe
  - Compromising xPU, PCIe-SC and its internal connection

- TCB size:
  - TVM: 3.1K Lines of Codes
  - PCIe-SC:
    - 218.6K ALUTs
    - 195.7K Registers
    - 630 BRAMs

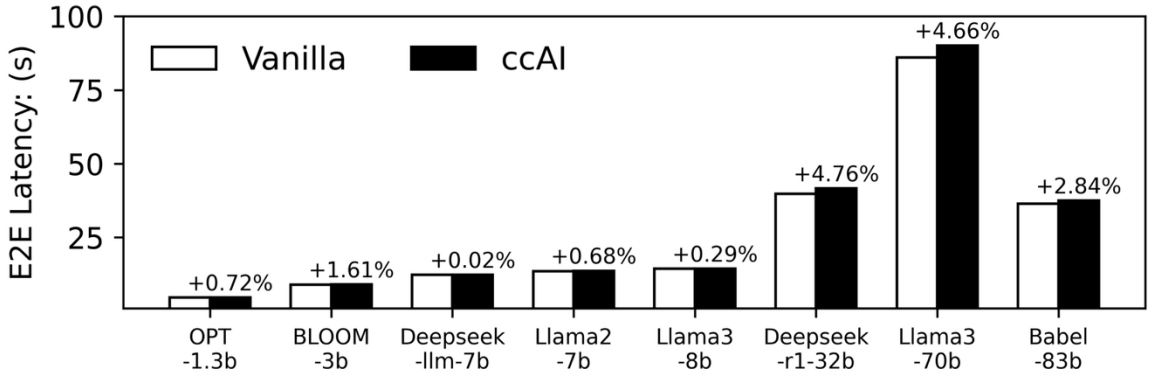
Components	LoC	ALUTs	Regs	BRAMs
TVM				
Adaptor	2.1K	–	–	–
Trust Modules	1.0K	–	–	–
PCIe-SC				
Packet Filter	–	11.3K	32.4K	310
Packet Handlers	–	175.5K	56.8K	72
HRoT-Blade	–	0	0	0
Others	–	31.5K	106.5K	248
Total	3.1K	218.6K	195.7K	630

Low Performance Overhead on Multi-xPUs/LLMs

- ccAI is compatible with **multi-xPUs**
  - NVIDIA A100, T4, RTX4090 GPU
  - Enflame S60 GPU
  - Tenstorrent N150d NPU

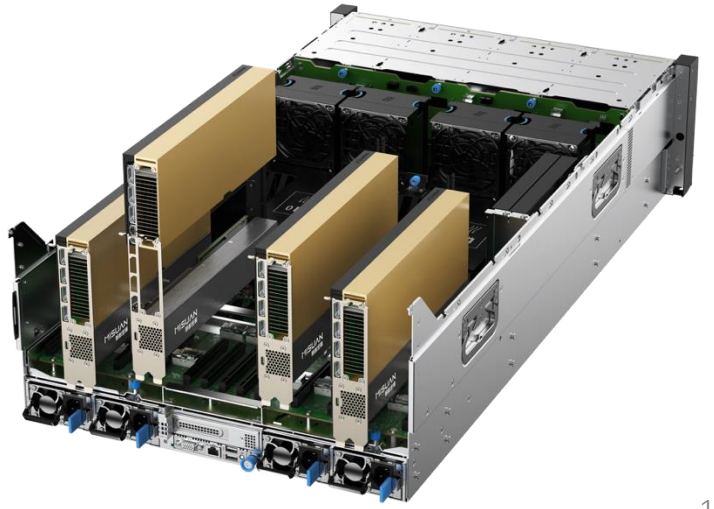


- ccAI is compatible with **multi-LLMs**
  - Deepseek-r1-32b (INT2)
  - Llama3-70b (INT2)
  - Babel-83b (INT2)



Conclusions

- **ccAI** provides heterogeneous clouds with confidential xPU-based AI computing
  - **No changes** on xPU application, xPU software, and xPU hardware device
  - **Bottom-layer (PCIe packet) protection** to ensure compatibility, Integrity, Isolation and Authenticity
  - **Low (0.05% - 5.67%)** performance overhead
- **ccAI** product is released!



SUSTech Southern University of Science and Technology
COMPASS Lab
COMPuter And System Security Lab

ANT GROUP

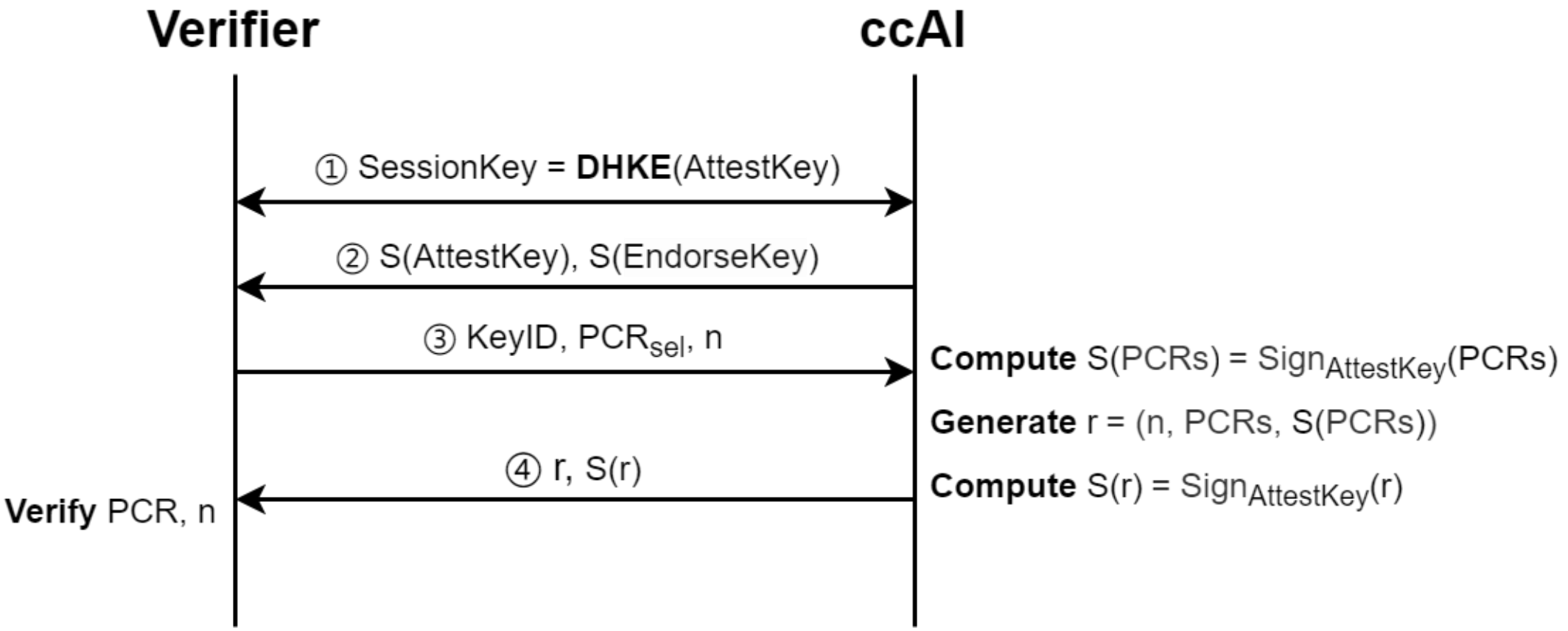
THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Thank You!

Contact presenter: 12150073@mail.sustech.edu.cn

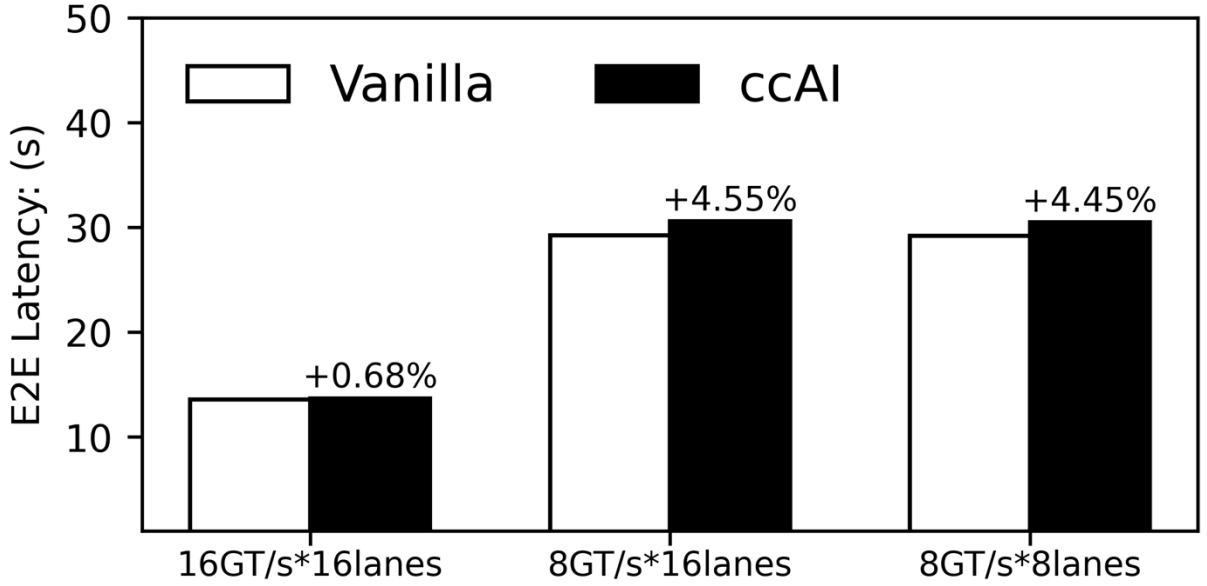
Backup Slides

Remote Attestation Workflow

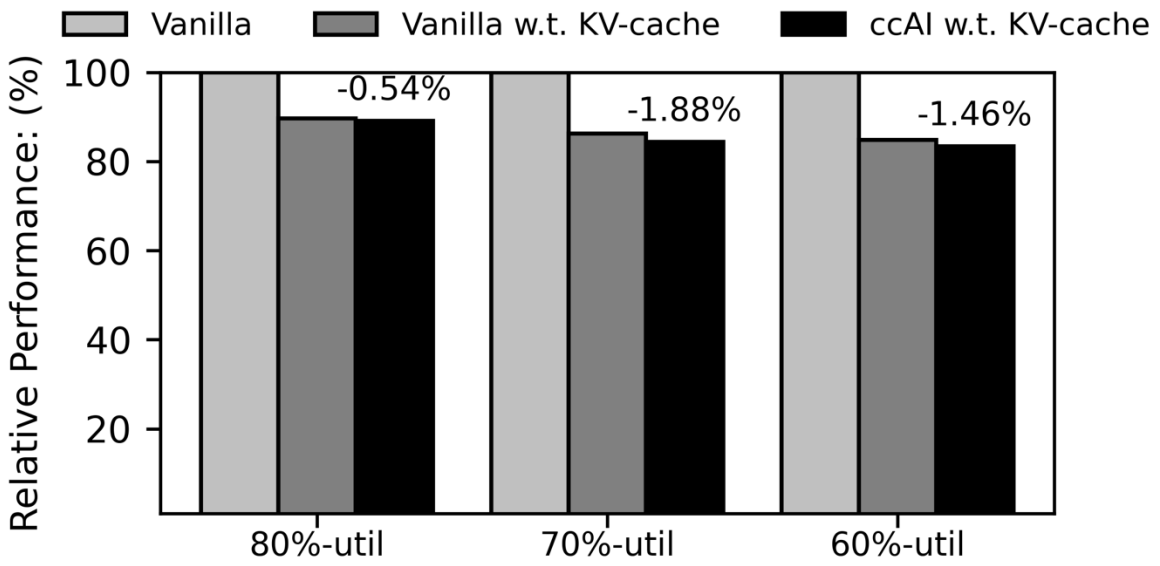


PCR: Platform Configuration Register in HRoT-Blade, used for generating attestation report

Other Performance Test



Limited PCIe bandwidth



Limited memory and KV-cache swap

ccAI vs PCIe Channel Encryption?

- CC-GPU requires PCIe channel encryption.
  - e.g., NVIDIA H100 GPUs

- Compared to PCIe Channel Encryption…
  - **PCIe 3.0** is OK for ccAI, no requirement for PCIe IDE (after PCIe 5.0)
  - **Pipeline** to optimize encryption time