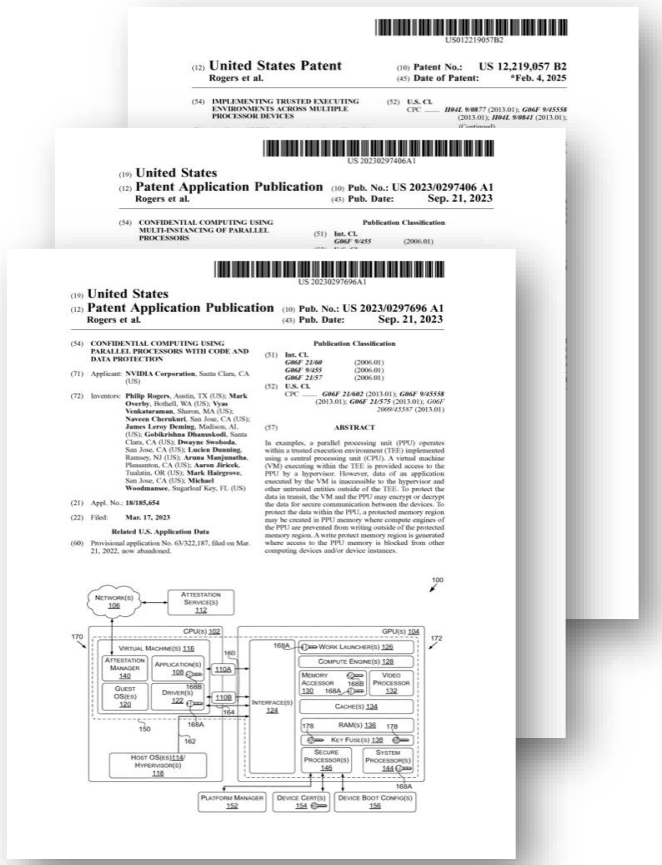
Ninth Annual Conference on Machine Learning and Systems (MLSys 2026)
Security and Privacy

Blueprint, Bootstrap, and Bridge: A Security Look at NVIDIA GPU Confidential Computing

Zhongshu Gu*, Enriquillo Valdez*, Salman Ahmed*, Julian Stephen*,
Michael Le*, Hani Jamjoom*, Shixuan Zhao+, Zhiqiang Lin+
IBM Research*, The Ohio State University+

NVIDIA GPU Confidential Computing

GPU-CC is critical for protecting sensitive AI workloads in the cloud

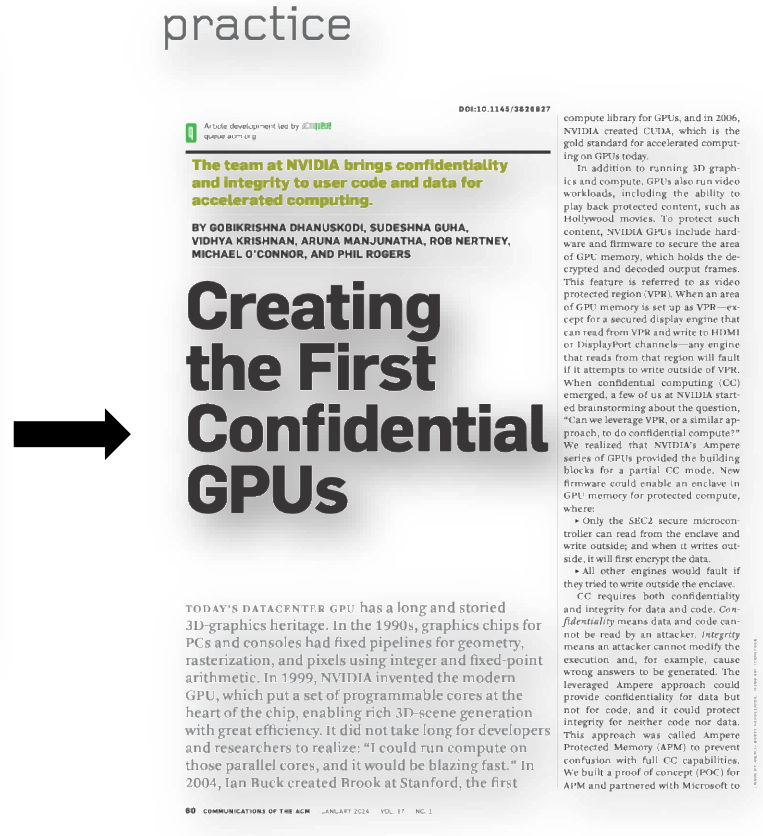


(12) **United States Patent**
Rogers et al.
(10) Patent No.: US 12,219,057 B2
(45) Date of Patent: *Feb. 4, 2025
(54) IMPLEMENTING TRUSTED EXECUTING ENVIRONMENTS ACROSS MULTIPLE PROCESSOR DEVICES

(19) United States
(12) Patent Application Publication
Rogers et al.
(10) Pub. No.: US 2023/0297406 A1
(43) Pub. Date: Sep. 21, 2023
(54) CONFIDENTIAL COMPUTING USING MULTI-INSTANCING OF PARALLEL PROCESSORS

(19) United States
(12) Patent Application Publication
Rogers et al.
(10) Pub. No.: US 2023/0297696 A1
(43) Pub. Date: Sep. 21, 2023
(54) CONFIDENTIAL COMPUTING USING PARALLEL PROCESSORS WITH CODE AND DATA PROTECTION
(57) ABSTRACT

Patents: filed in 2021 and 2023



practice

DOI:10.1145/3626827

The team at NVIDIA brings confidentiality and integrity to user code and data for accelerated computing.

BY GOBIKRISHNA DHANUSKODI, SUDESHNA GUHA, VIDHYA KRISHNAN, ARUNA MANJUNATHA, ROB NERTNEY, MICHAEL O'CONNOR, AND PHIL ROGERS

Creating the First Confidential GPUs

TODAY'S DATACENTER GPU has a long and storied 3D-graphics heritage. In the 1990s, graphics chips for PCs and consoles had fixed pipelines for geometry, rasterization, and pixels using integer and fixed-point arithmetic. In 1999, NVIDIA invented the modern GPU, which put a set of programmable cores at the heart of the chip, enabling rich 3D-scene generation with great efficiency. It did not take long for developers and researchers to realize: "I could run compute on those parallel cores, and it would be blazing fast." In 2004, Ian Buck created Brook at Stanford, the first compute library for GPUs, and in 2006, NVIDIA created CUDA, which is the gold standard for accelerated computing on GPUs today.

In addition to running 3D graphics and compute, GPUs also run video workloads, including the ability to play back protected content, such as Hollywood movies. To protect such content, NVIDIA GPUs include hardware and firmware to secure the area of GPU memory, which holds the decrypted and decoded output frames. This feature is referred to as video protected region (VPR). When an area of GPU memory is set up as VPR—except for a secured display engine that can read from VPR and write to HDMI or DisplayPort channels—any engine that reads from that region will fault if it attempts to write outside of VPR. When confidential computing (CC) emerged, a few of us at NVIDIA started brainstorming about the question, "Can we leverage VPR, or a similar approach, to do confidential compute?" We realized that NVIDIA's Ampere series of GPUs provided the building blocks for a partial CC mode. New firmware could enable an enclave in GPU memory for protected compute, where:

- Only the SEC2 secure microcontroller can read from the enclave and write outside; and when it writes outside, it will first encrypt the data.
- All other engines would fault if they tried to write outside the enclave.

CC requires both confidentiality and integrity for data and code. *Confidentiality* means data and code cannot be read by an attacker. *Integrity* means an attacker cannot modify the execution and, for example, cause wrong answers to be generated. The leveraged Ampere approach could provide confidentiality for data but not for code, and it could protect integrity for neither code nor data. This approach was called Ampere Protected Memory (APM) to prevent confusion with full CC capabilities. We built a proof of concept (POC) for APM and partnered with Microsoft to

60 COMMUNICATIONS OF THE ACM JANUARY 2024 VOL. 67 NO. 1

CACM Paper: 2024

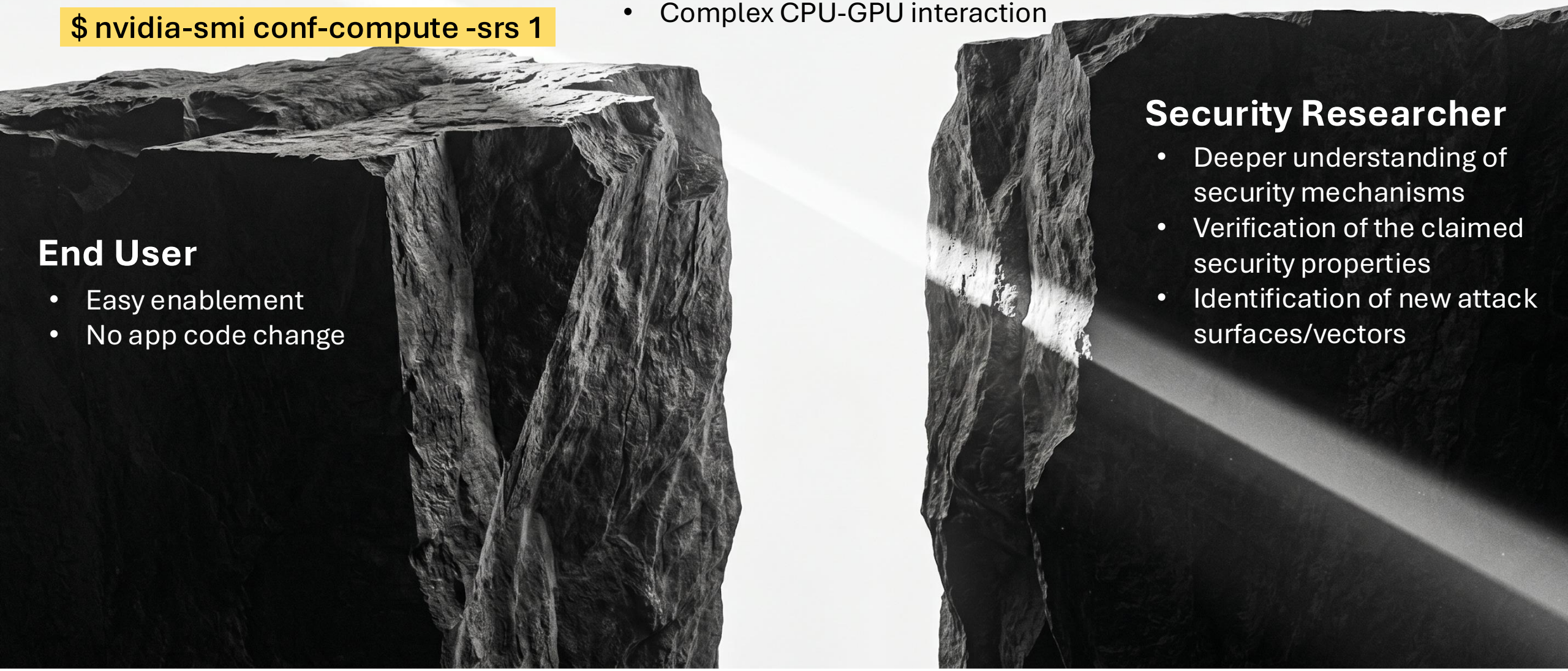
Hardware support since Hopper

Motivation

Knowledge Gap
Principles ↔ Mechanisms

- Proprietary components
- Undocumented functionalities
- Complex CPU-GPU interaction

`$ nvidia-smi conf-compute -srs 1`



End User

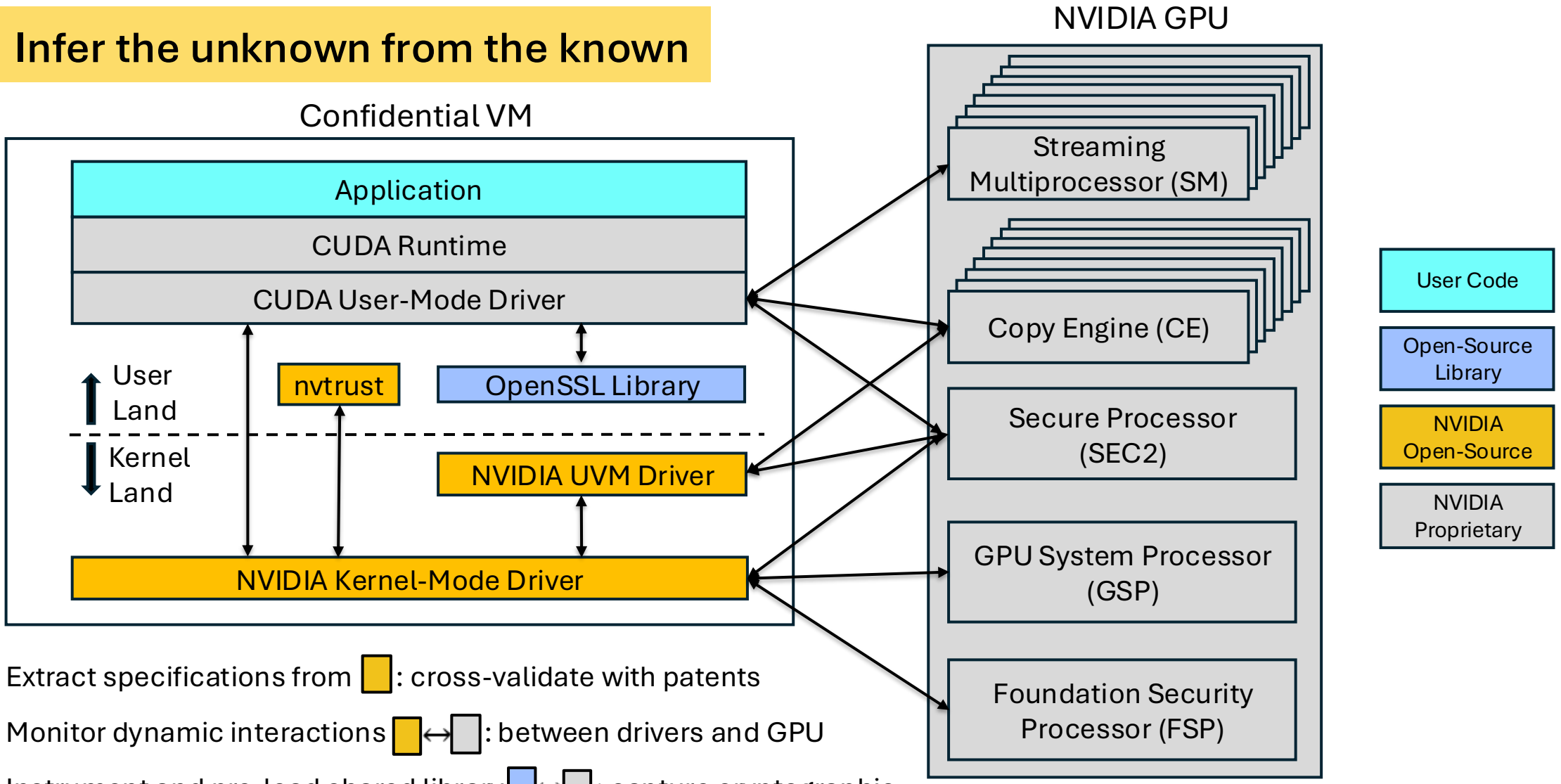
- Easy enablement
- No app code change

Security Researcher

- Deeper understanding of security mechanisms
- Verification of the claimed security properties
- Identification of new attack surfaces/vectors

Methodology

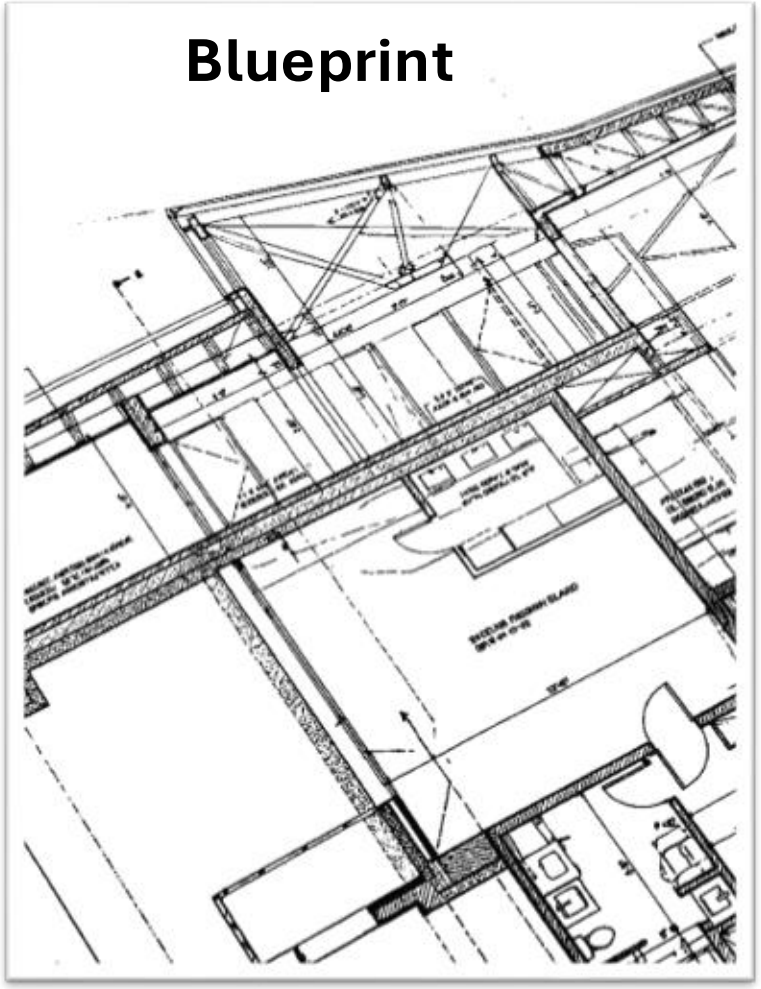
Infer the unknown from the known



- Extract specifications from : cross-validate with patents
- Monitor dynamic interactions : between drivers and GPU
- Instrument and pre-load shared library : capture cryptographic operations

Approach

Blueprint

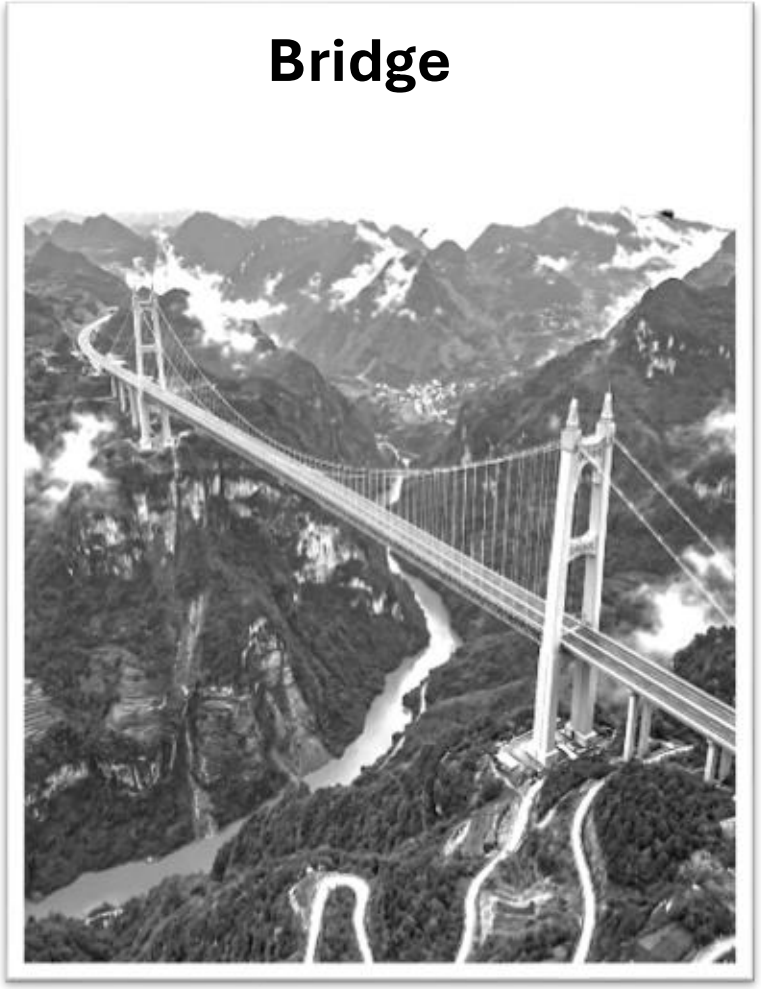


What are the key architectural components of GPU-CC?

Bootstrap

How is trust established between the CVM and the GPU?

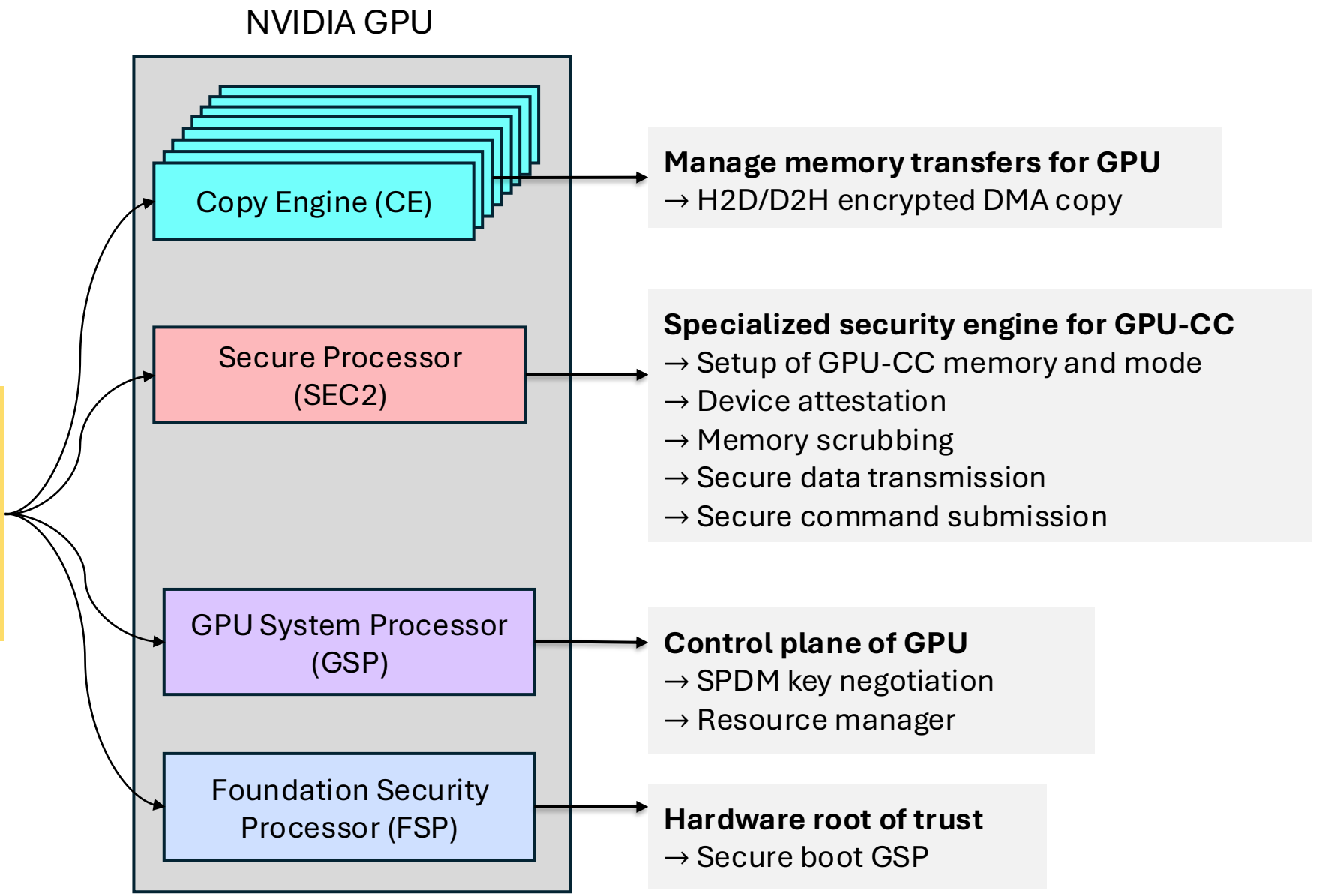
Bridge



Is data protected when crossing the untrusted interconnect?

Blueprint

GPU-CC functionalities are distributed across hardware engines



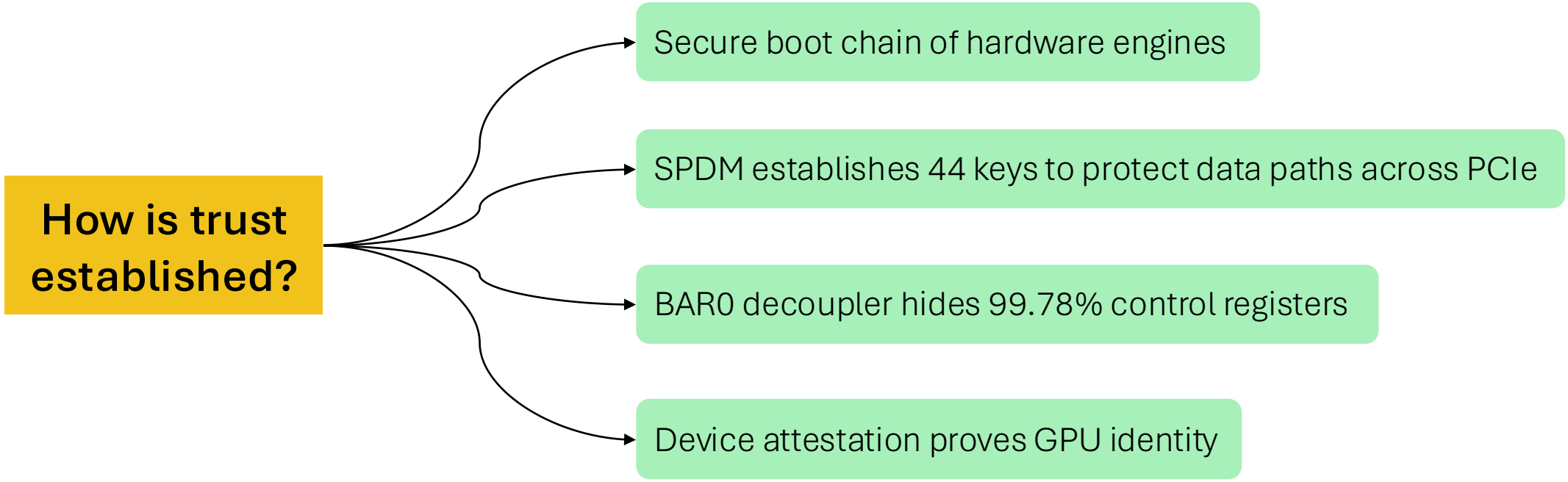
Manage memory transfers for GPU
→ H2D/D2H encrypted DMA copy

Specialized security engine for GPU-CC
→ Setup of GPU-CC memory and mode
→ Device attestation
→ Memory scrubbing
→ Secure data transmission
→ Secure command submission

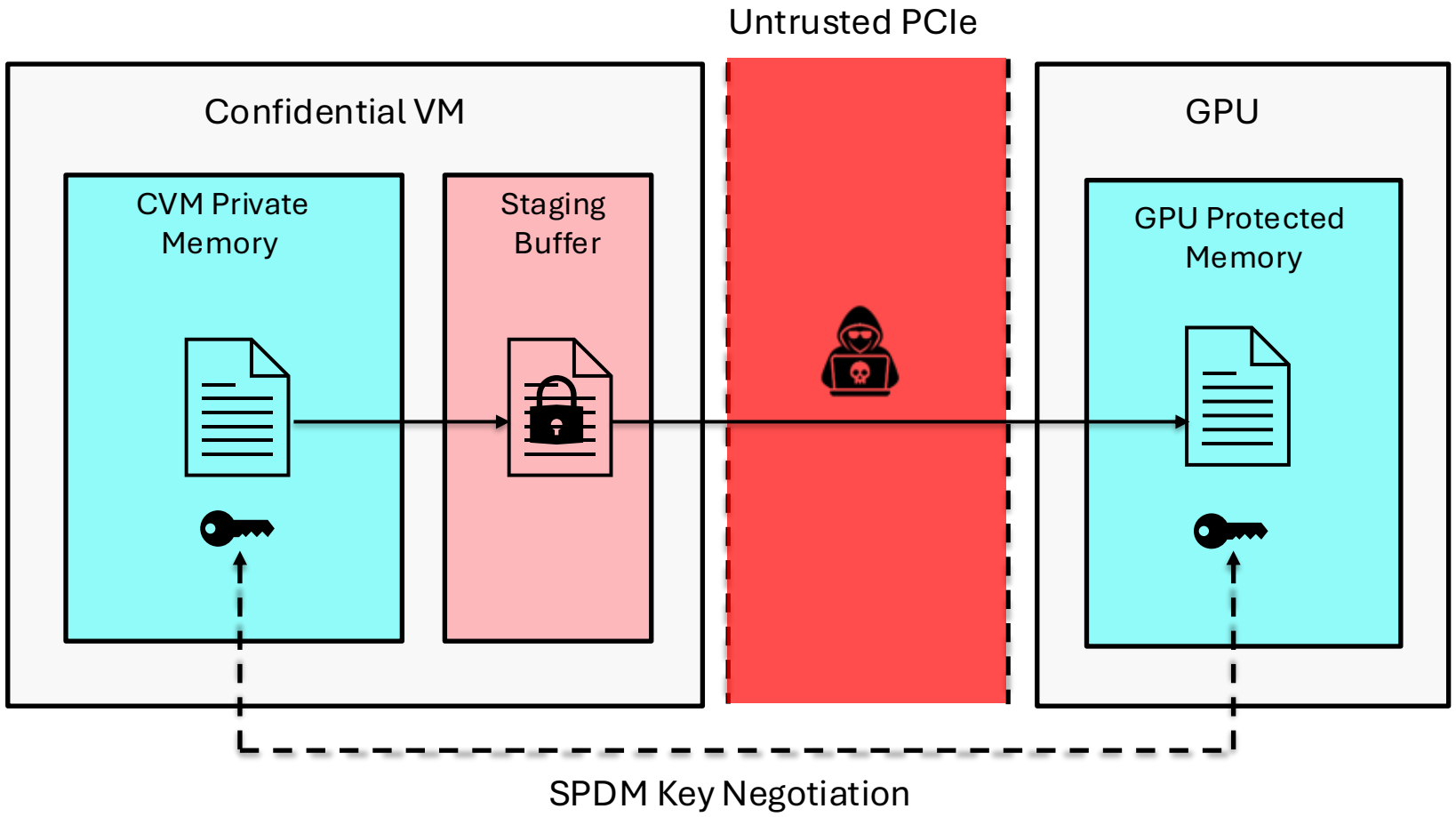
Control plane of GPU
→ SPDM key negotiation
→ Resource manager

Hardware root of trust
→ Secure boot GSP

Bootstrap


How is trust established?
Secure boot chain of hardware engines
SPDM establishes 44 keys to protect data paths across PCIe
BAR0 decoupler hides 99.78% control registers
Device attestation proves GPU identity

Bridge



Six Data Paths across PCIe

- CPU-GSP RPC
- CPU-GSP memory transfers
- GPU memory faults
- UVM operations
- Memory scrubbing command
- CUDA

Information exposure during data transmission

- RPC metadata
- Timing channels
- Command queue metadata
- Semaphore signals

Security Analysis on CPU-GSP RPC

CPU-GSP RPC:
- NVIDIA kernel-mode driver → GSP-RM API

Security Issue:
- RPC payload is encrypted, but RPC queue metadata are not encrypted

Track Data Pointers:
- Physical address table → queue header/element → queue metadata

Locate physical address table in a large memory space?
- Self-reference: first entry stores table's own address
- Scan memory for "`address == address[0]`"

Physical Address Table

(qemu) xp /129gx 0x16921e000

000000016921e000: 0x000000016921e000 0x000000016921f000
000000016921e010: 0x0000000169297000 0x0000000169300000
...

Queue Header

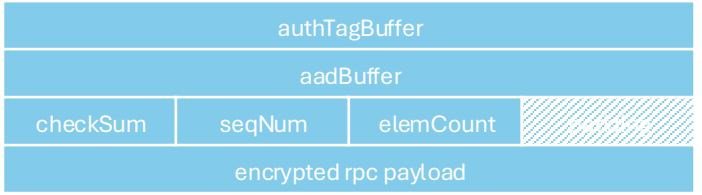
(qemu) xp /12wx 0x000000016921f000

000000016921f000: 0x00000000 0x00040000 0x00001000 0x0000003f
000000016921f010: 0x0000001e 0x00000001 0x00000020 0x00001000
000000016921f020: 0x00000023 0x00000000 0x00000000 0x00000000

Queue Element

(qemu) xp /20wx 0x0000000169297000

0000000169297000: 0xef1c9c0a 0x92f51983 0x0deb4ee8 0x2030226c
0000000169297010: 0x0000088a 0x00000000 0x00000000 0x00000000
0000000169297020: 0xef206d88 0x0000088a 0x00000001 0x00000000
0000000169297030: 0x7dcbfb21 0xc0fef69d 0x3382c0d9 0x9a5a2d6b
...



Conclusion

- NVIDIA GPU-CC is a major step forward to enable secure AI
- GPU-CC secures bulk data, but not always the context around it
- The proprietary components still challenge security analysis

We hope this work helps demystify NVIDIA GPU-CC and lays the foundation for future security research