SoK: Analysis of Accelerator TEE Designs

Chenxu Wang[1*†‡], Junjie Huang[1†], Yujun Liang[†], Xuanyao Peng[†¶], Yuqun Zhang[†],
Fengwei Zhang[†*✉], Jiannong Cao[‡], Hang Lu[¶], Rui Hou[§¶], Shoumeng Yan[‖✉], Tao Wei[‖] and Zhengyu He[‖]
[*]Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, China
[†]Department of Computer Science and Engineering, Southern University of Science and Technology, China
[‡]Department of Computing, The Hong Kong Polytechnic University, China
[§]State Key Laboratory of Cyberspace Security Defense, IIE, Chinese Academy of Sciences, China
[¶]University of Chinese Academy of Sciences, China,
[‖]Ant Group, China

***Abstract*—Accelerator trusted execution environment (TEE) is a popular technique that provides strong confidentiality, integrity, and isolation protection on sensitive data/code in accelerators. However, most studies are designed for a specific CPU or accelerator and thus lack generalizability. Recent TEE surveys partially summarize the threats and protections of accelerator computing, while they have yet to provide a guide to building an accelerator TEE and compare the pros and cons of their security solutions. In this paper, we provide a holistic analysis of accelerator TEEs over the years. We conclude a typical framework of building an accelerator TEE and summarize the widely-used attack vectors, ranging from software to physical attacks. Furthermore, we provide a systematization of accelerator TEE's three major security mechanisms: (1) access control, (2) memory encryption/decryption, and (3) attestation. For each aspect, we compare varied security solutions in existing studies and conclude their insights. Lastly, we analyze the factors that influence the TEE deployment on real-world platforms, especially on the trusted computing base (TCB) and compatibility issues.**

I. Introduction

Trusted execution environment (TEE) is a popular and widely used security mechanism to ensure data confidentiality and integrity on today's platforms. Over the past two decades, the industry and academy propose various TEE designs and hardware supports on both cloud platforms [1], [2] and mobile devices [3], supporting general and state-of-the-art requirements such as data storage [4]–[7], machine learning [8]–[10], and blockchain applications [11]–[13]. These designs provide isolated computing environments and secure CPU resources for users, protecting the sensitive data storage and computing from the untrusted software components.

Currently, extending TEEs from CPU to other accelerator devices, such as Graphics Processing Units (GPUs) [14]–[18], Neural Processing Units (NPUs) [19]–[22], Tensor Processing Units (TPUs) [23], Field Programmable Gate Arrays (FPGAs) [24]–[26], and other accelerators [27], is gradually popular. By using accelerator TEEs, users securely perform high-performance computing on their sensitive tasks and enjoy the benefits of strong confidentiality, integrity, and isolation. Nevertheless, there is no standard criterion for building an accelerator TEE. Studies propose a large number of accelerator TEE designs, ranging from large-scale clouds [28]–[30] to lightweight endpoints [31]–[33], from traditional Intel platforms [34], [35] to the new RISC-V architecture [36], [37]. Unfortunately, directly applying one accelerator TEE design to a different platform can be challenging due to the variance in CPU and accelerator architecture. Recent TEE surveys partially analyze security computing on GPU [8] and FPGA [38]. They also provide a security framework and concerns on CPU-side TEEs [39]–[41]. However, there is an absence of a systematic analysis of the accelerator TEE framework, potential threats to mainstream security solutions, and a discussion of the pros and cons of deploying these TEEs in real-world platforms.

To address this problem, we provide a holistic analysis of accelerator TEEs. Our analysis answers the primary question: **RQ1: What is the typical framework of building an accelerator TEE?** We survey the state-of-the-art accelerator TEEs in the past decade (detailed in §II). Based on this, we categorize accelerator TEEs into three types: (1) *Host-type*, which mainly protects accelerators with CPU-side software/firmware; (2) *Acc.-type*, which prefers to design their protection on accelerators and the connection I/O bus; and (3) *Mix-type*, which is a mixed design of these two types. We detail our categorization in §III. Despite the CPU architecture, accelerator devices, and platforms' variance, accelerator TEEs follow the aforementioned types to deploy security protection.

Since we categorize accelerator TEEs and summarize their design features, our study focuses on a further question: **RQ2: How to build an accelerator TEE with ensuring strong security on varied CPU/accelerator?** To answer this question, we summarize the widely-used attack vectors in the typical accelerator TEE framework (detailed in §IV). Attacks on accelerator TEEs can vary from CPU-side application to privileged hypervisor, from software to physical attacks. Based on the attack vectors, we summarize three major defense mechanisms against the powerful adversary: (1)

[1]Chenxu Wang and Junjie Huang are co-first authors.
✉Fengwei Zhang and Shoumeng Yan are corresponding authors.

Network and Distributed System Security (NDSS) Symposium 2026
23 February - 26 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.231424
www.ndss-symposium.org

access control, (2) memory encryption, and (3) attestation, then provide a detailed solution categorization for each mechanism. Besides categorization, we provide several insights for each mechanism. For access control, we focus on summarizing the solutions (and their combination) preference on the three types of accelerator TEE (i.e., *Host-type*, *Acc.-type*, and *Mix-type*). For memory encryption, we find that many studies lack such a security solution and cannot defend against physical threats. We analyze the significant overhead problems in existing memory encryption designs due to the traditional solution migration and inconsistent granularity. For attestation, we summarize the generic attestation workflow for accelerator TEE. Based on this, we find that most solutions lack the essential security supports to ensure attestation correctness. For the aforementioned security mechanisms, we detail our analysis and insights in §V, §VI, and §VII, respectively.

Lastly, for deploying accelerator TEEs on real-world devices, we focus on another question: **RQ3: What factors influence the accelerator TEE deployment on real-world platforms?** With the sharp increase in accelerator applications, however, accelerator TEEs have yet to be widely applied in real-world platforms. We consider two major aspects: large TCB and low compatibility. Specifically, we analyze the TCB size of the guest and system side for each accelerator TEE (detailed in §VIII) and provide a compatibility analysis (detailed in §IX). Our analysis shows that most accelerator TEEs have non-trivial TCB requirements and non-negligible compatibility issues. This influences the deployment of accelerator TEEs and should be carefully addressed in future TEE designs.

We summarize our contributions as follows:

- We summarize current design choices of accelerator TEEs into three main classifications: *Host-type*, *Acc.-type*, and *Mix-type* designs.
- We describe attack vectors and their capabilities for accelerator computing. This model benefits TEE designers in terms of defending against specific attacks.
- We summarize the mainstream solutions on access control. We also analyze the preference for solutions and their combination in existing studies.
- We categorize the solutions for memory encryption. We compare these solutions' pros and cons on granularity, security guarantee, and performance aspects.
- We analyze existing solutions in attestation, with a detailed attestation workflow and the lacking components of existing studies.
- We analyze the TCB size in existing studies. Our analysis indicates security concerns of increasing guest/system TCB in accelerator TEE designs.
- We comprehensively discuss compatibility issues across existing studies, especially in multi-type and plug-and-play support on software/hardware.

II. Methodology

We perform a literature review focusing on the research studies about accelerator computing protection with TEE technology. Following the state-of-the-art [38], [83], [84], we collect research studies from Google Scholar[1], a mainstream search engine that indexes most research studies from digital libraries (e.g., ACM Digital Library[2], IEEE Xplore[3], Arxiv[4]). To use this search engine, we first perform a search query with two sets of keywords — accelerator devices (e.g., `GPU`, `NPU`, `TPU`, `DPU`, `IPU`, `ASIC`, `FPGA` and `xPU`) and TEE technology (e.g., `TEE`, `Confidential Compute`, and architecture-specific TEEs such as `Arm TrustZone` and `Intel SGX`). This results in more than 20K studies, while our search stops at the 200th study — most studies after this are irrelevant to our study. For these studies, we consider two criteria to filter our target studies:

- Academic or industry studies should leverage at least one TEE-based security mechanism (e.g., Intel SGX [1], Arm CCA [85], or customized hardware [58]) to protect the computing environment of at least one accelerator.
- Academic studies should provide a detailed defense mechanism on the accelerator computing workflow rather than protecting I/O for generic devices.

These criteria help us filter the accelerator TEE studies. To

Table I: Overview of the analyzed accelerator TEEs.

Acc. TEE	Year	Pub.	Host CPU	Acc.	Src.	CPU TEE
Graviton [34]	2018	OSDI	Intel	GPU	○	●
HIX [33]	2019	ASPLOS	Intel	GPU	○	●
HETEE [28]	2020	S&P	Any[2]	General Acc.	○	○
TrustOre [42]	2020	CCS	Intel	FPGA Acc.	○	●
Telekine [43]	2020	NSDI	Intel	GPU	○	●
Ambassy [44]	2021	TMC	Arm	U-FPGA Acc.	○	●
CommonCounters [45]	2021	HPCA	Intel	GPU	○	●
CURE [36]	2021	USENIX	RISC-V	U-Acc.	○	●
PSSM [46]	2021	ICS	Intel	GPU	○	●
SGX-FPGA [35]	2021	DAC	Intel	FPGA Acc.	○	●
Cronus [47]	2022	MICRO	Arm	General Acc.	●	●
GuardNN [48]	2022	DAC	Any[2]	DNN Acc.	○	○
LEAP [49]	2022	TMC	Arm	U-GPU	○	●
LITE [50]	2022	ICS	Intel/AMD	GPU	○	●
MGX [51]	2022	ISCA	Intel	DNN Acc.	○	●
ShEF [52]	2022	ASPLOS	Any[2]	FPGA Acc.	●	○
StrongBox [31]	2022	CCS	Arm	U-GPU	●	●
TNPU [53]	2022	HPCA	Intel	U-NPU	○	●
SHM [54]	2022	HPCA	Intel	GPU	○	●
RME-DA [55]	2023	(Industry)	Arm	General Acc.	○	●
SEV-TIO [56]	2023	(Industry)	AMD	General Acc.	○	●
TDX Connect [57]	2023	(Industry)	Intel	General Acc.	○	●
H100 [58]	2023	(Industry)	Intel/AMD/Arm	GPU	○	●
AccShield [59]	2023	DAC	Intel/AMD	TPU	○	●
AvaGPU [32]	2023	CCS	Arm	U-GPU	●	●
GR-T [60]	2023	EuroSys	Arm	U-GPU	●	●
Honeycomb [61]	2023	OSDI	AMD	GPU	○	●
ITX [62]	2023	ATC	Any[2]	IPU	○	○
MyTEE [63]	2023	NDSS	Arm	U-GPU	●	●
Plutus [64]	2023	HPCA	Any[2]	GPU	○	○
SAGE [65]	2023	ATC	Intel	GPU	●	●
Securator [66]	2023	HPCA	Any[2]	U-NPU	○	○
ACAI [67]	2024	USENIX	Arm	General Acc.	●	●
CAGE [68]	2024	NDSS	Arm	U-GPU	●	●
Dhar et al. [30]	2024	ACSAC	Any[1]	General Acc.	○	●
HyperTEE [69]	2024	MICRO	RISC-V	U-DNN Acc.	○	●
Na et al. [70]	2024	HPCA	Intel	GPU	○	●
Salus-GPU [71]	2024	HPCA	Any[2]	GPU	○	○
Salus-FPGA [72]	2024	ASPLOS	Intel	FPGA Acc.	○	●
sIOPMP [73]	2024	ASPLOS	RISC-V	U-DNN Acc.	○	●
sNPU [37]	2024	ISCA	RISC-V	U-NPU	○	●
SrcTEE [74]	2024	TC	Arm	U-FPGA Acc.	○	●
T-Edge [75]	2024	ACSAC	Arm	U-FPGA Acc.	●	●
TensorTEE [76]	2024	ASPLOS	Intel	NPU	○	●
ASGARD [77]	2025	NDSS	Arm	U-NPU	●	●
ccAI [78]	2025	MICRO	Any[1]	General Acc.	○	●
GuardAIn [29]	2025	S&P	Any[2]	NPU	○	○
PipeLLM [79]	2025	ASPLOS	Intel/AMD	GPU	○	●
Portal [80]	2025	S&P	Arm	U-GPU	○	●
SeDA [81]	2025	DAC	Any[2]	DNN Acc.	●	○
XpuTEE [82]	2025	TOCS	Intel	GPU	○	●

[1]It relies on CPU TEE with memory encryption support.
[2]It adapts any CPU architecture with/without CPU TEE.

[1]https://scholar.google.com
[2]https://dl.acm.org
[3]https://ieeexplore.ieee.org/
[4]https://arxiv.org

extend our paper collection, we also perform forward/backward searching on these studies (i.e., searching studies that refer to and are referred). Overall, we analyze 51 studies, which are selected from top-level conferences and mainstream industry designs, and report them in Table I. We extensively discuss our observations and relevant insights as follows.

A. Motivation

Accelerator TEE study is popular but not systematic. In the past three years, accelerator TEEs have gradually attracted the public's attention and have sharply increased. As shown in Table I, 41/51 studies, including four industry studies [55]–[58], have been proposed since 2022. However, compared with the CPU TEE with a long history (e.g., Arm TrustZone [3] in 2004), the formal study of accelerator TEE only begins in 2018 [34]. In addition, designs of accelerator TEEs heavily rely on CPU TEEs (42/51 studies) and have yet to summarize a unique design framework. Based on this, our paper needs to address a primary problem: **RQ1: What is the typical framework of building an accelerator TEE?**

Analyzing accelerator TEE is challenging. Studies on accelerator TEEs are exploring various CPU-accelerator combinations. Currently, we find more than ten CPU-accelerator combinations in accelerator TEE designs — Accelerator TEEs protect various accelerators (e.g., GPU and NPU) that run with different accelerator computing workflows and are equipped on various CPU hosts (e.g., Arm, Intel, and RISC-V). Most accelerator TEEs prefer to be compatible with x86-based (i.e., Intel and AMD) platforms (32/51 studies) and support GPU computing (30/51 studies). This is because x86-based platforms and GPUs are the mainstream heterogeneous systems for high-performance computing. In addition, we observe that studies for Arm-based accelerator TEEs prefer to support unified-memory accelerators (11 studies). The major reason is that most Arm-based platforms are endpoints and support embedded accelerators. For RISC-V accelerator TEEs, 3/4 studies prefer to support configurable NPU or DNN accelerators. Due to the complex accelerator TEE implementation, we propose the second question: **RQ2: How to build an accelerator TEE while ensuring strong security on varied CPU/accelerator?**

Accelerator TEEs have yet to be widely deployed. Clouds and endpoints gradually support multiple types of accelerators to work together on various accelerator tasks. However, most accelerator TEEs target specific CPU-accelerator combinations (43/51 studies). We observe that 40/51 studies support a specific CPU architecture, and 43/51 studies are only designed for one type of accelerator device. Considering that different CPUs and accelerators are largely varied in security and functionality, we worry that most accelerator TEEs have yet to support a generic accelerator computing environment. Worse, only a few accelerator TEEs (12/51 studies) release their source code. It is challenging to migrate accelerator TEE research to different platforms. We raise the third research question: **RQ3: What factors influence the accelerator TEE deployment on real-world platforms?**

III. Overview

A. System Model

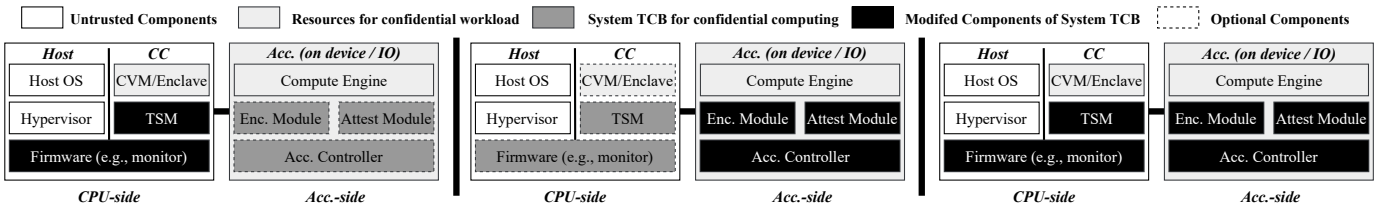
Accelerator TEE definition and requirements. Compared to traditional CPU TEEs (e.g., Intel SGX [1] and Arm TrustZone [3]), accelerator TEEs extend protection to accelerator devices while ensuring the security goals of confidentiality, integrity, and authenticity. During the accelerator task preparation, computation, and termination phases, TEE-related components must secure two key elements: (1) Accelerator workloads, including input/output data, model parameters, task code (e.g., AI models), page tables, and other confidential metadata; and (2) the accelerator environment, such as the hardware status of both the CPU and accelerator. Beyond these primary security guarantees, accelerator TEEs can also provide additional security support (e.g., protecting entire software stacks) while minimizing influences on accelerators (e.g., maintaining compatibility and performance).

CPU side. Accelerator TEEs, which are deployed on the mainstream architectures (e.g., x86, Arm, and RISC-V), follow a generic architecture layout on the CPU side. Generally, the CPU side software consists of three major components: (1) a host running a hypervisor and host OS, (2) a CC (i.e., confidential computing) environment running a TSM (i.e., TEE Security Manager [86]) and several confidential virtual machines (CVMs) or enclaves [87] [2] [88], and (3) a firmware layer running the highest privilege software (e.g., a monitor [88] [3]). The host and CC environment are strictly isolated with security hardware support (e.g., Arm TZASC in TrustZone [3]), which can be configured by monitor or TSM.

Accelerator side. We summarize the mainstream accelerator or extension hardware design for accelerator TEEs. First, an accelerator delegated to different workloads must equip a compute engine (including computing units, registers, caches, and other computing resources) to process workloads. Moreover, accelerators with security supports additionally provide three components: (1) the encryption module (e.g., AES-GCM engine in Microsoft ITX [62]) for security communication and memory protection, (2) the attestation module for attestation, such as the Hardware Root-of-Trust (HRoT) in NVIDIA H100 [58], and (3) the security controller (e.g., Task Scheduler in GuardAIn [29]) for TEE management. For accelerator memory, several accelerators (e.g., NVIDIA [58] and AMD GPUs [15]) own a physically isolated memory (e.g., GDDRx [45]) while others (e.g., Arm GPUs [89]) share the main memory with CPU and other peripherals.

B. Accelerator TEE Categorization

Categorizing the surveyed accelerator TEEs can be challenging since they are implemented on different platforms (i.e., various CPU-accelerator combinations) with different security mechanisms, and require different levels of software/hardware changes. Nevertheless, based on our observations, *these accelerator TEEs still require system control components to coordinate different security mechanisms for accelerator confidential computing*. Thus, based on the position of the modified



(a) Host-type accelerator TEEs (b) Acc.-type accelerator TEEs (c) Mixed-type accelerator TEEs

Figure 1: Architecture overview of the *Host-type*, *Acc.-type*, and *Mix-type* accelerator TEEs.

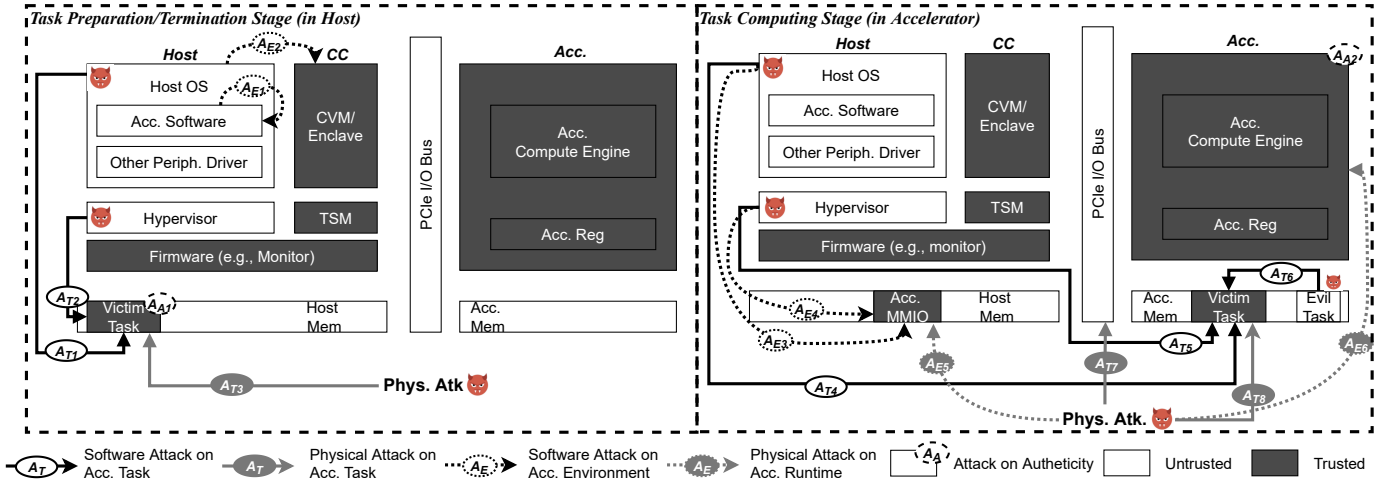


Figure 2: Attack vectors of accelerator computing.

system control components, we classify these accelerator TEEs into three categories: *Host-type* designs, *Acc.-type* designs, and *Mix-type* designs. Figure 1 shows a brief architecture overview of these three designs. Moreover, we provide detailed components of different accelerator TEE types in Table II. In this table, we categorize the selected studies into three types (i.e., *Host-type*, *Acc.-type*, and *Mix-type*) and show the security components related to accelerator TEE (i.e., CVM/enclave, TSM, firmware, encryption module, attestation module, and accelerator controller).

***Host-type* designs.** As shown in Figure 1(a), *Host-type* designs modify privileged CPU-side system control components (e.g., the TSM such as RMM [67] [80] or secure monitor [31] [68]) to control accelerator TEEs and leverage CPU-assisted protection mechanisms (e.g., Intel SGX [1] and Arm TrustZone [3]) to isolate, attest, and secure the accelerator computing environment. During the trusted accelerator computing, the key role of the system control component is to ensure the data path and interaction between the TEE on the host and the accelerator hardware, preventing data leakage from untrusted software (e.g., OS and hypervisor) and devices. Since CPU-side hosts generally implement mature security mechanisms, most *Host-type* designs directly leverage these mechanisms in access control (e.g., Intel EPC [1], Arm TZASC [90]/GPC [91]) and memory encryption (e.g., Intel TME [87], AMD SME [2]), without additional hardware changes on host or accelerator devices. However, several designs still implement customized security primitives (e.g., bus filters in CURE [36] and SGX extensions in HIX [33]).

***Acc.-type* designs.** Compared to the *Host-type* designs, the *Acc.-type* designs migrate the accelerator TEE control logic into accelerator (e.g., the command processor [34] [43] in GPUs) or extension hardware (e.g., the security controller [28] [30] between host and accelerator), instead of preserving it in CPU side (see Figure 1(b)). This accelerator controller receives and controls data/command communication from external devices (e.g., the untrusted host [28] [29] or peripherals), manages TEE on accelerator and coordinate security operations (e.g., memory protection via encryption module [53] [51] and environment verification via attest module [58] [62]). The security of the *Acc.-type* controller is ensured by hardware isolation from the CPU host instead of the privilege software isolation. Thus, most *Acc.-type* designs unavoidably require hardware modification on accelerator board or extension security hardware, but do not involve host-side hardware changes. These features enable CPU TEE to work orthogonally with *Acc.-type* designs, such as NVIDIA H100 [58], which supports VM-level CPU TEE [87] [2] [88].

***Mix-type* designs.** Combining the former two, the *Mix-type* design collaborates the system control components at the CPU and accelerator side to control the accelerator TEE (see Figure 1(c)). In this case, privileged software on the CPU side guarantees the confidentiality and integrity of the communication data, and physical hardware on the accelerator side guarantees the security of the computational data. In this case, privileged software on the CPU side guarantees the confidentiality and integrity of accelerator task submission and termination. In contrast, security hardware on the accelerator

Table II: Design overview of the *Host-type*, *Acc.-type* and *Mix-type* accelerator TEEs.

Acc. TEE Type	CPU-side			Acc.-side modules (in Acc./Board/Ext. IO)		
	CVM/Enclave	TSM	Firmware	Enc. Module	Attest Module	Acc. Controller
***Host-type* Acc. TEEs**						
ACAI [67]	Arm CCA	RMM	Monitor	PCIe IDE(Acc.)	HRoT(Acc.)	-
ASGARD [77]	Arm TrustZone	S-Hyp	Monitor	-	HRoT(Acc.)	-
AvaGPU [32]	Arm TrustZone	-	Monitor	-	-	-
Cronus [47]	Arm TrustZone	S-Hyp	Monitor	-	HRoT(Acc.)	-
CURE [36]	RISC-V Customized	-	M-Monitor	-	-	-
CAGE [68]	Arm CCA	RMM	Monitor	-	-	-
GR-T [60]	Arm TrustZone	-	Monitor	-	-	-
Honeycomb [61]	AMD SEV-SNP	SVSM	SEV-firmware	-	-	-
HIX [33]	Intel SGX	-	SGX-firmware	-	HRoT(Acc.)	-
HyperTEE [69]	RISC-V Customized	-	M-Monitor	-	-	-
LEAP [49]	Arm TrustZone	-	Monitor	-	-	-
MyTEE [63]	Arm TrustZone	-	Monitor	-	-	-
Portal [80]	Arm CCA	RMM	Monitor	-	-	-
sIOPMP [73]	RISC-V Penglai	-	M-Monitor	-	-	-
StrongBox [31]	Arm TrustZone	-	Monitor	-	-	-
XpuTEE [82]	Intel TDX/SGX	-	VMX root	-	-	-
***Acc.-type* Acc. TEEs**						
AccShield [59]	Intel TDX/AMD SEV	TDX module/SVSM	TDX/SEV-firmware	AES-GCM Engine(Board)	HRoT(Board)	Security Manager(Board)
Ambassy [44]	Arm TrustZone	-	Monitor	AES Cores(Acc.)	-	Acc. Controller(Acc.)
CommonCounters [45]	Intel SGX	-	SGX-firmware	Opti-Enc. Engine(Acc.)	-	Command Processor(Acc.)
ccAI [78]	-[1]	-	-	AES-GCM Engine(Ext.IO)	HRoT(Ext.IO)	PCIe-SC(Ext.IO)
Dhar et al. [30]	-[1]	-	-	AES-GCM Engine(Ext.IO)	HRoT(Ext.IO)	Security Controller(Ext.IO)
GuardAIn [29]	-	-	-	AES-GCM Engine(Acc.)	HRoT(Acc.)	Task Scheduler(Acc.)
GuardNN [48]	-	-	-	Opti-Enc. Engine(Acc.)	-	Micro-controller(Acc.)
Graviton [34]	Intel SGX	-	SGX-firmware	AuthEnc/Dec. kernel(Acc.)	HRoT(Acc.)	Command Processor(Acc.)
HETEE [28]	-	-	-	AES-GCM Engine(Ext.IO)	HRoT(Ext.IO)	Security Controller(Ext.IO)
ITX [62]	-	-	-	AES-GCM Engine(Board)	CCU(Board)	ICU(Board)
LITE [50]	Intel TDX/AMD SEV	TDX module/SVSM	TDX/SEV-firmware	Enc. kernel&Spec. HW(Acc.)	-	Acc. Controller(Acc.)
MGX [51]	Intel SGX	-	SGX-firmware	Opti-Enc. Engine(Acc.)	HRoT(Acc.)	Control Processor(Acc.)
Na et al. [70]	Intel SGX	-	-	Opti-Enc. Engine(Acc.)	HRoT(Acc.)	Command Processor(Acc.)
NVIDIA H100 [58]	Intel TDX/AMD SEV/Arm CCA	TDX module/SVSM/RMM	TDX/SEV-firmware/Monitor	AES-GCM Engine(Acc.)	HRoT(Acc.)	Acc. Controller(Acc.)
PipeLLM [79]	Intel TDX/AMD SEV/Arm CCA	TDX module/SVSM/RMM	TDX/SEV-firmware/Monitor	AES-GCM Engine(Acc.)	HRoT(Acc.)	Acc. Controller(Acc.)
Plutus [64]	-	-	-	Opti-Enc. Engine(Acc.)	-	Memory Controller(Acc.)
PSSM [46]	Intel SGX	-	SGX-firmware	Opti-Enc. Engine(Acc.)	-	Command Processor(Acc.)
Salus-FPGA [72]	Intel SGX	-	SGX-firmware	AES-GCM Engine(Acc.)	HRoT(Acc.)	SM Controller(Acc.)
Salus-GPU [71]	-	-	-	Opti-Enc. Engine(Acc.)	-	Memory Controller(Acc.)
Securator [66]	-	-	-	Opti-Enc. Engine(Acc.)	-	Security Module(Acc.)
SeDA [81]	-	-	-	Opti-Enc. Engine(Acc.)	-	Memory Controller(Acc.)
ShEF [52]	-	-	-	Engine set(Board)	HRoT(Board)	Shield(Board)
SAGE [65]	Intel SGX	-	SGX-firmware	AuthEnc/Dec. kernel(Acc.)	Kernel(Acc.)	Kernel Caller(Acc.)
SGX-FPGA [35]	Intel SGX	-	SGX-firmware	Enc. Engine(Acc.)	PUF(Acc.)	FPGA Secure Monitor(Acc.)
SHM [54]	Intel SGX	-	SGX-firmware	Opti-Enc. Engine(Acc.)	-	Command Processor(Acc.)
SrcTEE [74]	Arm TrustZone	-	Monitor	AES-GCM Engine(Board)	PUF(Board)	Config. Sec. Unit(Board)
Telekine [43]	Intel SGX	-	SGX-firmware	AuthEnc/Dec. kernel(Acc.)	HRoT(Acc.)	Command Processor(Acc.)
T-edge [75]	Arm TrustZone	-	Monitor	Enc. Engine(Acc.)	HRoT(Acc.)	Acc. Controller(Acc.)
TrustOre [42]	Intel SGX	-	SGX-firmware	AES-GCM Engine(Acc.)	Attester(Acc.)	TrustMod(Acc.)
TNPU [53]	Intel SGX	-	SGX-firmware	Opti-Enc. Engine(Acc.)	-	Memory Controller(Acc.)
TensorTEE [76]	Intel SGX	-	SGX-firmware	Opti-Enc. Engine(Acc.)	-	Memory Controller(Acc.)
***Mix-type* Acc. TEEs**						
Arm RME-DA [55]	Arm CCA	RMM	Monitor	PCIe IDE(Acc.)	HRoT(Acc.)	DSM(Acc.)
AMD SEV-TIO [56]	AMD SEV	SVSM	SEV-firmware	PCIe IDE(Acc.)	HRoT(Acc.)	DSM(Acc.)
Intel TDX Connect [57]	Intel TDX	TDX module	TDX-firmware	PCIe IDE(Acc.)	HRoT(Acc.)	DSM(Acc.)
sNPU [37]	RISC-V Penglai	-	M-Monitor	-	-	Isolator, Guard(Acc.)

[1]It relies on CPU TEE with memory encryption support; DSM: Device Security Manager, a centralized security module in TDISP-compliant device.; Acc.: The corresponding module is built into the accelerator; Board: Security hardware integrated with accelerators on the same board (e.g., Shield [52]); Ext. IO: External IO security hardware, separate from the accelerator on the same board. (e.g., Security Controller [28], [30]).

side ensures the security of task computation. Currently, there are limited *Mix-type* design efforts (4/51 studies). In these efforts, the TEE Device Interface Security Protocol (TDISP) [86] proposed by PCI-SIG has attracted widespread attention [55]–[57]. Specifically, the Device Security Manager (DSM) on the accelerator side manages one or more TEE Device Interfaces (TDIs) that can be securely assigned to CVMs. The TDISP control path between the DSM and TSM is protected by the trusted channel (i.e., Security Protocol and Data Model [92]), and the data path is protected with the PCI Integrity and Data Encryption (IDE) protocol [93].

Answer to RQ1. As summarized in §III, the framework of accelerator TEEs includes three major types (*Host-type*, *Acc.-type*, and *Mix-type*), in which system components can be flexibly configured to meet TEE requirements. Nevertheless, when TEE designers and vendors select a TEE type for implementation, we provide actionable guidelines as follows: First, for most cloud providers (e.g., Aliyun [94]) using commercial accelerators, we recommend designing a *Host-type* TEE. Cloud providers can secure accelerator workloads (and even software) within CVMs and implement accelerator environment/hardware protection by modifying the TSM and firmware. Additionally, security components can be easily implemented and upgraded via software patches. Second, for accelerator manufacturers (e.g., NVIDIA [95] and Xilinx [24]), we suggest designing an *Acc.-type* TEE by integrating controllers, custom encryption IP cores, and attestation modules into the accelerator hardware. With authorization from TEE users, manufacturers can leverage third-party CVMs to secure their software stacks. Third, for TEE designers/vendors with independent design capabilities (e.g., Apple [18], Huawei [96]), we recommend a *Mix-type* TEE. In this design, TEE designers/vendors can combine modifications to CPU-side TSM/firmware and accelerator hardware to achieve robust, custom-built accelerator protection.

IV. Attack Vectors

As shown in Figure 2, we categorize attacks to accelerator computing into three types: (1) attack on accelerator task (marked as A_T) including the task code, data, metadata (e.g., task buffer pointer), page tables, and other task resources stored in host or accelerator memory, and (2) attack on accelerator environment (marked as A_E) including the accelerator software, Memory Mapped I/O (MMIO) registers, and hardware device, and (3) attack on authenticity (marked as A_A). All three types of attack are executed whether the victim task is in the host (i.e., task preparation/termination stage) or in the accelerator (i.e., task computing stage).

A. Attacks in Task Preparation/Termination

In the task preparation or termination stage, the victim task (with input data or execution results) is stored in host-side memory and has yet to interact with the accelerator. We elaborate on the three types of attacks in this stage as follows.

To leak the sensitive data, the adversary may directly attack the victim task via the compromised host OS (A_{T1}) or hypervisor (A_{T2}). To achieve this, the adversary aims to

Table III: Security solutions comparison among accelerator TEEs.

	Solutions	CVM/Enclave: Acc. Workload	CVM/Enclave: Acc. Driver	TSM	Firmware	CPU HW	Bus	Enc. Module	Attest Module	Acc. Controller	Attacks in Task Preparation/Termination: A_{T1}	A_{T2}	A_{T3}	A_{E1}	A_{E2}	A_{A1}	Attacks in Task Computing: A_{T4}	A_{T5}	A_{T6}	A_{T7}	A_{T8}	A_{E3}	A_{E4}	A_{E5}	A_{E6}	A_{A2}
Access Control	S_{AC1}	✓									●	●	○	○	○	○	◐	◐	○	○	○	○	○	○	○	○
	S_{AC2}	✓	✓								●	●	○	●	○	○	◐	◐	●	○	○	○	○	○	○	○
	S_{AC3}			✓							●	○	○	●	◐	○	●	○	●	○	○	●	○	○	○	○
	S_{AC4}				✓						●	●	○	●	●	○	●	●	●	○	○	●	●	○	○	○
	S_{AC5}						✓				○	○	○	○	○	○	●	●	○	◐	○	●	●	◐	○	○
	S_{AC6}									✓	○	○	○	○	○	○	●	●	●	◐	◐	◐	◐	◐	●	○
Mem Enc	S_{ME1}	✓	✓			✓					●	●	●	○	○	○	○	○	○	◐	○	●	●	●	○	○
	S_{ME2}							✓			○	○	○	○	○	○	●	●	●	◐	●	○	○	○	○	○
	S_{ME3}						✓				○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○
Attestation	S_{AT1}					✓					○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	◐
	S_{AT2}	✓	✓					✓			○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○
	S_{AT3}								✓		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●
	S_{AT4}									✓	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●

S_{AC1}: Access control based on CPU TEE but not protecting accelerator software stacks. S_{AC2}: Access control based on CPU TEE and protecting accelerator software stacks. S_{AC3}: Access control based on TEE manager (e.g., RMM) or secure hypervisor.
S_{AC4}: Access control based on firmware (e.g., monitor). S_{AC5}: Access control based on IO bus (e.g., CPU bus filter and PCIe filter). S_{AC6}: Access control based on accelerator hardware (e.g., security controller [28]) and authenticated kernel (e.g., kernel caller [65]).
S_{ME1}: Enc/Dec based on CPU TEE. S_{ME2}: Enc/Dec based on encryption engine/kernel of accelerator. S_{ME3}: Enc/Dec based on IO bus (e.g., TDISP).
S_{AT1}: Attestation based on CPU TEE with HRoT. S_{AT2}: Attestation based on software. S_{AT3}: Attestation based on accelerator with HRoT. S_{AT4}: Attestation based on extension hardware with HRoT.
●, ◐, ○: Completely defend against this attack; Defend against this attack in partial scenarios; Unable to defend against this attack.

compromise the confidentiality or integrity of the victim's task. By compromising privileged software, adversaries can directly access the sensitive data (e.g., input, parameters, and execution results) or code of the victim task. Besides compromising task confidentiality, the adversary can threaten task integrity. The adversary can achieve this by replacing the input data of the victim task or by changing or injecting malicious code into the victim tasks. The adversary can achieve the attacks above by using the controlled Direct Memory Access (DMA)-capable peripherals. In addition, the adversary may modify the page table and metadata (e.g., buffer pointer and page table pointer) of the victim task, misleading the accelerator to access incorrect data buffers or compute with malicious codes. Besides the software attacks, the adversary may launch the aforementioned attacks with physical assists (A_{T3}), such as extracting sensitive data from host DRAM or replaying the outdated execution results.

The adversary may compromise the accelerator runtime with two types of attack: (1) interfering with the functionality of accelerator software (A_{E1}) and (2) performing Iago attacks (A_{E2}). For A_{E1} attack, the adversary may compromise the task scheduling to provide the incorrect task execution order, such as shuffling the execution order of tasks, arbitrarily replaying task execution, dropping tasks in confidential applications, or terminating the victim task ahead. Also, the adversary may compromise the memory management of accelerator software, such as interfering with the memory allocation for task buffers and page tables, tampering with the page table mapping (e.g., duplicated mapping or mapping to an unprotected region) in the victim task. For A_{E2} attack, the adversary may provide an incorrect value to TEE to mislead the accelerator protection. For instance, the adversary may give an incorrect page table or metadata address of the victim task, misleading the TEE to protect an unexpected region. The adversary may also replay the outdated data in the register to mislead the accelerator computation.

In addition, the adversary may compromise the authenticity of the victim task (A_{A1}). In the preparation stage of accelerator TEEs, the adversary may provide CVM/enclaves with incorrect task resources (i.e., data and task model).

B. Attacks in Task Computing

In task computing stage, the victim task is moved from the host to the accelerator-side memory. In this stage, accelerator software usually interacts with accelerators via MMIO. We elaborate on the three types of attacks in this stage.

The adversary may attempt to directly access or tamper with the victim task in the accelerator with the host OS (A_{T4}) or the hypervisor (A_{T5}) privilege. Although such an attack can be challenging on dedicated-memory accelerators (e.g., NVIDIA GPUs [14]) due to hardware isolation, it is more feasible to attack unified-memory accelerators (e.g., Arm GPUs [16]) that share the same memory with the host. Moreover, the adversary may abuse the accelerator TEEs and compromise the task isolation inside the accelerator (A_{T6}). For instance, the adversary may concurrently run an evil task with a victim task on the same accelerator. This evil task can monitor the execution of the victim task or directly access the victim task with modified page table mapping. A physical adversary can directly access the I/O bus (such as the PCIe I/O bus, shown in A_{T7}) or accelerator-side memory (A_{T8}) to leak or tamper with the sensitive task. Moreover, the adversary may tamper with the metadata or page table contents to achieve the same attacks as those on the host.

The adversary may access the accelerator MMIO on the host via OS (A_{E3}) or hypervisor (A_{E4}), compromising the accelerator runtime during the accelerator TEE computing. Through MMIO, the adversary may change the register values of critical accelerator registers (e.g., page table base registers). Also, since several registers are delegated to control the accelerator computing, the adversary may send illegal execution commands to these registers to early execute/terminate the victim task, or perform unauthorized data transmission between the host and accelerator. For a physical adversary, she can execute the attacks above through tampering with MMIO (A_{E5}), or even physically compromising the accelerator hardware (A_{E6}).

Lastly, the adversary may compromise the authenticity of the accelerator (A_{A2}). Specifically, she may route the victim tasks to an unexpected or emulated device, monitoring the computing of the victim tasks.

Table IV: Access control solutions preference among accelerator TEEs.

Scenario	Deployment Features	Access Control Solution						Acc. TEE Type			Specific Mechanism
		S_{AC1}	S_{AC2}	S_{AC3}	S_{AC4}	S_{AC5}	S_{AC6}	*Host-type*	*Acc.-type*	*Mix-type*	
CPU-Discrete Acc. (e.g., cloud)	CPU with Any TEE (e.g., TDX/SEV for Multi-tenants)	●	○	○	○	○	●		[34] [51] [70] [45] [46] [72] [35] [54] [42] [43] [76]		Intel SGX, Hardware-based Acc. Controller(Acc.)
	Re-programmed Acc. (e.g., FPGA for designer, Hopper GPU of NVIDIA)								[65]		Intel SGX, Kernel Caller(Acc.)
		○	●	○	○	○	●		[58] [50] [79]		CVM with MEE, NVIDIA CC hardware-supported (Acc.)
									[59]		Intel TDX/AMD SEV, Security Manager(Board)
	Plug-and-play Link (e.g., PCIe-based Sec. HW)	○	●	○	○	●	●			[56] [57] [55]	TDISP
		○	●	○	○	●	○	[33]			Intel SGX, PCIe Root Complex
									[30] [78]		Any CPU TEE, Security Controller(Ext. IO)
	CPU with Specific TEE (e.g., VMX root of Intel) Legacy Acc. (e.g., A100)	●	○	●	○	○	○	[61]			AMD SEV, SVSM
								[82]			Intel SGX/TDX, VMX root
		○	●	●	○	○	○	[47]			Arm TrustZone, S-Hyp
		○	●	●	●	○	○	[67]			Arm CCA, RMM, Monitor
	Legacy CPU (e.g., w/o CPU TEE) Re-programmed Acc.	○	○	○	○	○	●		[64] [48] [71] [29] [81]		Acc. Controller(Acc.)
									[62] [52]		Integrated Security Hardware(Board)
	Legacy CPU-Acc.	○	○	○	○	●	○		[28]		PCIe-based Security Controller(Ext. IO)
	Preference	14/34	12/34	4/34	1/34	6/34	**26/34**	5/34	**26/34**	3/34	**Mainstream solution combination:** $S_{AC1/2} + S_{AC5/6}$
Integrated CPU-Acc. (e.g., edge)	Platform with specific sec. HW (e.g., TZASC/GPC in Arm) or modified privilege SW (e.g., S-Hyp/ trusted firmware in Arm)	●	○	○	●	○	○	[68]			Arm CCA, Monitor
								[31] [63]			Arm TrustZone/OP-TEE, Monitor
		○	●	●	○	○	○	[77]			Arm TrustZone, S-Hyp
		○	●	●	●	○	○	[80]			Arm CCA, RMM, Monitor
		○	●	○	●	○	○	[60] [32] [49]			Arm TrustZone, Monitor
	Re-programmed HW (e.g., RISC-V/ FPGA-based DNN Acc.)	○	●	○	●	●	○	[36] [69]			Customized RISC-V TEE, M-Monitor, CPU IO Filter
		●	○	○	●	●	○	[73]			RISC-V Penglai, M-Monitor, CPU IO Filter
		●	○	○	●	○	●			[37]	RISC-V Penglai, M-Monitor, Isolator, Guard(Acc.)
		○	●	○	○	○	●		[53]		Intel SGX, Memory Controller(Acc.)
									[75] [44] [74]		OP-TEE, Acc. Controller(Acc.)
		○	○	○	○	○	●		[66]		Acc. Controller(Acc.)
	Preference	5/17	11/17	2/17	**11/17**	3/17	6/17	**11/17**	5/17	1/17	**Mainstream solution combination:** $S_{AC1/2} + S_{AC4}$

V. Access Control

A. Solutions for Access Control

S_{AC1}: TEE without accelerator driver. Studies can reuse the CPU-side TEE, including heavyweight CVMs (e.g., CVMs in AMD SEV [61], Intel TDX [82], and Arm CCA [68]) or lightweight enclaves (e.g., enclaves in SGX [42], [65], [97] or RISC-V [37], [73], or OP-TEE in Arm [31], [63]), to protect the accelerator workloads with sensitive data/code in both task preparation/termination and computing stage. The workloads are stored in a secure memory with software- or hardware-assisted isolation, restricting unauthorized access from Host OS (A_{T1}) and hypervisor (A_{T2}). Moreover, for unified-memory accelerators that share the same memory with the host, this solution additionally protects tasks from the same adversary during the task computing stage (i.e., A_{T4} and A_{T5}).

S_{AC2}: TEE with accelerator driver. Besides protecting the accelerator workloads, studies can further extend their CPU-side TEE protection to software stacks of varied accelerators (e.g., GPUs [33], [36], [80], NPUs [53], or generic types of accelerators [55]–[57]), including the accelerator driver and user-layer libraries (e.g., CUDA [98] or OpenCL [99]). With TEE protection, the accelerator software can securely manage tasks and device status. This solution defends against the attacks in S_{AC1}. Additionally, it addresses the threats to accelerator software (A_{E1}) and illegal access (A_{T6}).

S_{AC3}: Hypervisor-based access control. Studies can leverage a hypervisor-layer software, such as a TSM (e.g., AMD SVSM [56], [61], Intel TDX module [57], and Arm secure hypervisor [47] and RMM [55], [67], [80]) to defend against attacks from a low-privileged adversary, such as Host OS or other CVM/enclaves. This type of solution is mainly achieved by configuring hypervisor-layer access control mechanisms, such as Stage-2 translation in MMU/IOMMU [100]. In the task preparation/termination stage, this solution prevents the OS from accessing victim tasks in host memory (A_{T1}) and compromising accelerator software (A_{E1}). Moreover, it can verify the exchanged data between the host and TEE to partially mitigate Iago attacks (A_{E2}). In the task computing stage, this solution prevents accessing victim tasks in the accelerator ($A_{T4,6}$) or accessing MMIO (A_{E3}).

S_{AC4}: Firmware-based access control. Studies can configure the highest-privilege firmware (e.g., Monitor [31], [63], [68] in Arm, and M-mode Monitor [36], [37], [73] in RISC-V) to manage access control against the compromised OS, illegal CVM/enclaves, and hypervisor via MMU, or even malicious peripherals via IOMMU. This is typically supported by an isolated and hardware-assisted security primitive, such as TZASC [90] or TZPC [101] in Arm TrustZone, GPC [91] in Arm CCA, and PMP [102] in RISC-V. This solution covers previous attack vectors (i.e., $A_{T1,2,4,5,6}$, $A_{E1,2,3}$) and additionally prevents the hypervisor's access to the task in the accelerator (A_{T5}) and MMIO (A_{E4}).

S_{AC5}: Access control in IO Bus. Studies may customize access control in the bus connection between the CPU-side host and accelerator hardware, such as customizing the IO bus filter [73], adding a CPU bus filter [36], and monitoring PCIe switch [28]. To achieve this, studies may change the bus configuration, or introduce additional security hardware. This solution focuses on the attacks in task computing, effectively addressing the unauthorized access to tasks in accelerator ($A_{T4,5}$) and filtering illegal MMIO configuration ($A_{E3,4}$). Moreover, the bus protection mechanism can filter the malicious MMIO configuration (A_{E5}) and data leakage on the bus (A_{T7}) from the physical adversary.

S_{AC6}: Access control in accelerator. Studies can provide accelerator with a security controller, such as a customized security module in accelerator (e.g., command/control processor [34], [43], [58]) to provide a maximal access control

guarantee in task computing stage. It effectively addresses the data leakage from CPU-side adversary and the evil tasks ($A_{T4,5,6}$) and partially mitigates the malicious MMIO status configuration ($A_{T3,4,5}$). Moreover, protection on the accelerator can defend against physical tampering of the accelerator memory (A_{T8}) and other device components (A_{E6}).

B. Insights on Access Control

IAC1: Deployment scenarios drive multi-solution combination. Access control is an essential security solution in accelerator TEE design. Table III shows that higher-privilege solutions (e.g., $S_{AC4,5,6}$) address unmitigated threats in lower-privilege ones (e.g., $S_{AC1,2,3}$). However, existing studies typically combine multiple solutions instead of relying on a single option. A key reason lies in the deployment characteristics of different scenarios (detailed in Table IV). For cloud-equipped accelerator TEEs, they usually implement on Intel- and AMD-based clouds (with unmodifiable firmware) and connect to discrete accelerators via external interfaces (e.g., PCIe). This makes the combination between CVM/enclave ($S_{AC1,2}$) and accelerator-side defense ($S_{AC5,6}$) — such as the hardware firewalls for NVIDIA H100 [58] — the mainstream choice. Such combination, however, can be constrained by the accelerator's programmability, motivating studies (e.g., XpuTEE [82]) to modify VMX Root to support legacy accelerators. Additionally, when the CPU lacks TEE support and the accelerator is non-programmable, efforts finally shift to modify PCIe I/O (e.g., HETEE [28]). For endpoint accelerator TEEs, the CPU and accelerators are typically integrated into Arm/RISC-V endpoint platforms. Studies tend to modify high-privilege software (e.g., secure hypervisor) and leverage existing security hardware (e.g., Arm TZASC or RISC-V PMP in monitor) to secure accelerators [31], [32], [49], [60], [68], [77], [80]. Overall, we recommend that TEE designers select hybrid access control solutions based on their specific scenarios.

IAC2: CPU-side TEE is not necessary for accelerator protection. Accelerator TEEs do not necessarily require CVM/enclave to secure accelerator software, or even the workload and environment. Instead, studies can integrate security checks/protection with privileged software or hardware within accelerator's workflow (e.g., $S_{AC3,4,5,6}$). As shown in Table IV, the number of studies based on S_{AC1} and S_{AC2} is comparable — 19 and 23 studies, respectively — and meanwhile 9 studies do not rely on TEE-based protections. A primary reason is that accelerator software can manipulate workloads (e.g., allocating memory) without accessing their exact contents. Thus, accelerator TEEs can implement full encryption for sensitive contents and optionally expose non-confidential information (e.g., MMIO register values, page table, metadata) to the driver. Furthermore, removing accelerator software from CVM/enclave can effectively reduce their TCB size (will analyze in **ITC1** and **ITC2**). Currently, S_{AC2} is only indispensable in specific cases, such as accelerator TEE with unmodifiable firmware/TSM/hardware or unchangeable software stacks.

IAC3: Overreliance on firmware-based solutions is infeasible. Accelerator TEEs cannot naively rely on firmware-based access control (S_{AC4}) to mitigate most attack vectors. A key reason is the significant granularity limitations of hardware-assisted security primitives. Table V illustrates these limitations for configurable security primitives (e.g., Arm TZASC [103], Arm GPC [85] and RISC-V PMP [104]) widely used in accelerator TEEs [31], [37], [63], [67], [73]. Specifically, most of them lack granularity support in two critical aspects: (1) limited and coarse-grained isolation, and (2) limited control over diverse access permission attributes — both of which are essential for MMUs in commercial accelerators. Consequently, overreliance on S_{AC4}-based access control can severely disrupt the accelerator's functionality. This leads studies to reuse fine-grained access control mechanisms from TSMs or CVMs/enclaves.

Table V: Granularity comparison among security primitives.

	Arm TZASC	Arm GPC	RISC-V PMP	Addr Trans.
Solution Type	S_{AC4}	S_{AC4}	S_{AC4}	$S_{AC1,2,3}$
Minimal Granularity	32KB	4KB	4Byte	4KB
Configurable Regions	Limited	Non-limited	Limited	Non-limited
Read/Write Distinction	Supported	Not Supported	Supported	Supported
Execute Permission	Not Supported	Not Supported	Supported	Supported
PAN/PXN Permission	Not Supported	Not Supported	Not Supported	Supported
Studies Examples	[31], [60]	[67], [68], [80]	[37], [73]	[31], [32], [47], [67]

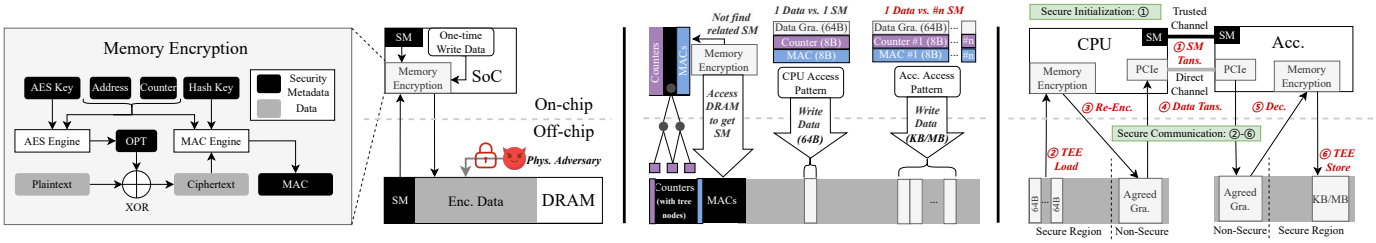
VI. Memory Encryption

A. Solutions for Memory Encryption

S_{ME1}**: CPU TEE-based memory encryption.** Studies can reuse the CPU TEE-side software encryption API or memory encryption engines (e.g., Intel TME [87] in [57], AMD SME [2] in [56], [61], Arm MEC [85] in [67], [100]) to protect sensitive data/code in the CPU-side off-chip memory during the preparation stage. These sensitive workloads are encrypted in memory to restrict privileged adversaries (i.e., A_{T2} and A_{T2}) and physical adversaries (i.e., A_{T3}) from accessing plaintext. Moreover, for unified-memory accelerators that share the same memory with the host, this solution protects tasks from the same adversaries during the task computing stage (i.e., $A_{T4,5,6,7,8}$).

S_{ME2}**: Accelerator kernel/hardware-based memory encryption.** Besides CPU-side encryption, studies can provide cryptographic support to the accelerator-side memory. They achieve this by delivering authorized en/decryption kernels [34], [50], [65] to the accelerator, or design hardware-based en/decryption engine [58], [59]. This solution effectively prevents physical adversaries (i.e., A_{T8}) from accessing data, and mitigates data leaks caused by CPU-side privileged adversaries (i.e., A_{T4} and A_{T5}) and malicious tasks (i.e., A_{T6}). To prevent physical adversaries from accessing confidential data (i.e., A_{T7}) on the external PCIe bus, this solution collaborates with S_{ME1} for encrypted data/code transmission.

S_{ME3}**: IO Bus-based memory encryption.** Lastly, studies may deploy memory encryption mechanisms on the bus between the CPU and the accelerator. When transmitting accelerator tasks with sensitive data/code, this solution carefully encrypt the transmission packets between the CPU and accelerator devices. To achieve this, studies may apply an



(a) Basic memory encryption against physical adversary

(b) Comparison of access patterns under traditional memory encryption

(c) Data transfer protocol under incompatible encryption granularity

Figure 3: Memory encryption workflows of accelerator TEEs.

Table VI: Security implications of missing memory encryption.

Scenarios	Victim	Physical Threats	Missing Memory Encryption (S_{ME}) → Consequence	Influenced Studies
CPU-Discrete Acc. (e.g., cloud)	Plug-and-play Host Memory (e.g., DDRx)	A_{T3}, A_{E5}	Missing CPU-based encryption (S_{ME1}) → Data/code/metadata/PTEs in plaintext on host memory are vulnerable to physical access/tampering (e.g., cold-boot attacks)	[28], [48] [47], [60]
	3D-stacked Acc. Memory (e.g., HBM)	-	Missing any memory encryption → Minimal physical threats	[34], [42], [43] [52], [58] [61], [62], [79] [71]
	On-board Acc. Memory (e.g., GDDRx/LPDDRx)	A_{T8}	Missing Acc.-based encryption (S_{ME2}) → Data/code/metadata/PTEs in plaintext on acc. memory are vulnerable to physical access/tampering (e.g., probing attacks)	[28], [33], [35], [47], [55]–[57], [60] [30], [65], [67], [72], [78], [82]
	Plug-and-play Link (e.g., PCIe/CXL)	A_{T7}	Missing IO-based (S_{ME3}) or CPU-Acc. encryption ($S_{ME1,2}$) → Physical access/tamper/replay packets in plaintext on the link (e.g., replay attacks).	[33], [47], [48], [82]
Integrated CPU-Acc. (e.g., edge)	On-board/Plug-and-play Shared Memory (e.g., LPDDRx/DDRx)	$A_{T3,8}, A_{E5}$	Missing CPU-Acc.-based encryption ($S_{ME1,2}$) → Data/code/metadata/PTEs in plaintext on shared memory are vulnerable to physical access/tampering	[31], [32], [36], [49], [63], [68], [80] [37], [44], [68], [73]–[75], [77], [80]

Missing de/encryption engine (e.g., AES) for confidentiality → Direct access to plaintext data
Missing integrity check engine (e.g., Message Authentication Code, MAC) for integrity → Tampering with plaintext/ciphertext data
Missing number used once (e.g., counter/integrity tree [105]) for freshness → Replay attacks

Table VII: Comparison of security metadata for accelerator TEE memory protection.

Solution	Studies	Freshness (via counter)					Integrity (via MAC)			
		Data Gra.	Counter Gra.	Space[1]	Storage	Protection	Data Gra.	MAC Gra.	Space[1]	Storage
Fine-gra. (Traditional MEE)	[1]	Block(64B)	Normal(56bit)	CBTL	Off-chip	MT	Block(64B)	Normal(56bit)	HBTL	Off-chip
Coarse-gra.	[53] [51] [48]	Tile(MB)	SW-aware(64bit)	CTL	On-chip	-	Block(64/512B)	Normal(64bit)	HBTL	Off-chip
	[76]	Tile(MB)	SW-aware(56bit)	CTL	On-chip	-	Tile(MB)	SW-aware(56bit)	HTL	Off-chip
	[66]	Layer(MB)	SW-aware(64bit)	CL	On-chip	-	Layer(MB)	SW-aware(32B)	HL	On-chip
Muti-gra.	[81]	Tile(MB)	SW-aware(64bit)	CTL	On-chip	-	Block,Layer, Model	SW-aware	[H, HBTL]	On-chip(Layer,Model) Off-chip(Block)
	[64] [46] [71]	Block(128B) Sector(32B)	Major(32bit) Minor(3/7/8bit)	CBTL	Off-chip(Block)	BMT(Block)	Block(128B)	Normal (64/56/32bit)	HBTL	Off-chip
	[45]	Segment(KB) Block(128B)	Common(32bit) Normal(64bit)	[CSTL, CBTL]	On-chip(Segment) Off-chip(Block)	- (Segment) BMT(Block)	Block(128B)	Normal(64bit)	HBTL	Off-chip
	[54]	R-Region(KB) Sector(32B)	Major(32bit) Minor(32bit)	[CPTL, CBTL]	On-chip(Region) Off-chip(Block)	- (R-Region) BMT(Block)	Page(4KB) Block(64B)	Normal(64bit)	[HPTL, HBTL]	On-chip(Page) Off-chip(Block)

[1]: A large language model (block size * BLT) as the memory protection object;
SW-aware: counters based on software-detected tensor-granularity scheme; Segment: uniformly updated segments (e.g., 128KB [45]); R-Region: Read-only region (e.g., 16KB [54]); BMT: Bonsai Merkle Tree;
H: MAC size; C: Counter size; B: nums of blocks per tile; P: nums of pages per tile; S: nums of segments per tile; T: nums of tiles per layer; L: Total nums of layers;

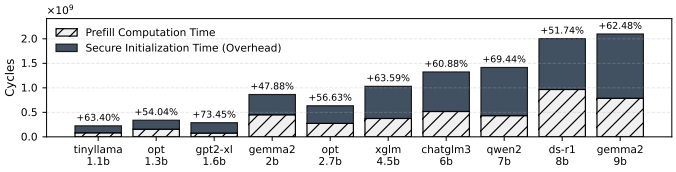
external encryption engine [28], [59] or modifying the I/O bus [93]. This solution benefits from the fact that it requires no Memory Encryption Engines (MEEs) on the host or accelerator. Meanwhile, it provides the security capabilities of S_{ME1} and S_{ME2} and defend against the same adversaries (i.e.,$A_{T2,3,4,5,6,7,8}$).

B. Insights on Memory Encryption

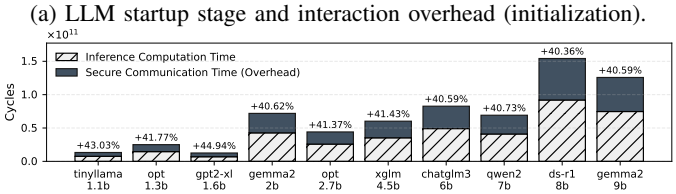
IME1: Physical adversaries are underestimated. The lack of memory encryption tailored to different deployment scenarios can expose systems to severe attack vectors. As shown in Table VI, cloud accelerators are typically equipped with high-capacity discrete memory and connect to the host via PCIe. In this case, several accelerators [106]–[108] do not physically integrate their memory with the accelerator chip — thus, they require accelerator-based memory encryption (S_{ME2}) to mitigate threats from physical adversaries. Besides accelerator devices, the PCIe link necessitates IO-based encryption (S_{ME3}) or co-encryption ($S_{ME1,2}$); otherwise, transmitted plaintext packets are vulnerable to interception, tampering, and replay attacks. For endpoint devices, the CPU and accelerator typically share the same memory [16], [18]. If the endpoint memory is compromised, the accelerator TEE remains vulnerable to physical attacks [109] and requires co-encryption between the CPU and accelerator ($S_{ME1,2}$). Currently, more than half of the studies (30/51 studies in Table VI) are impacted by various attacks targeting accelerator environments and workloads. Therefore, we recommend that TEE designers systematically analyze physical threats across different scenarios to implement appropriate memory encryption solutions.

IME2: Improper-grained security metadata significantly increases overhead. When designing memory encryption for a specific accelerator, naively reusing CPU TEE's or other accelerator TEE's solution can introduce large performance overhead. Specifically, CPUs access memory at the cache line granularity (64B), while accelerators (e.g., NPUs and GPUs) access memory in large blocks (KB or MB) of contiguous data. Thus, when reading from or writing to memory, CPU requires a single memory access to retrieve security metadata, while accelerator needs frequent accesses to multiple sets of

security metadata (see Figure 3(b)). To mitigate this overhead, studies (11/51 studies in Table VII) focus on tailoring memory encryption solutions for accelerators. They aim to scale up metadata granularity to reduce the frequency of creating metadata entries. Moreover, NPU TEEs [53] [51] [48] [81] [66] [76] adopt coarse-grained memory encryption solutions since NPUs usually support neural network computing scenarios (which access memory in tiles or layers). However, such coarse-grained solutions are unsuitable for GPU TEEs due to two reasons. First, GPUs feature sector memory [46] and global memory [54] with diverse memory access granularities. Second, most GPU data blocks (e.g., intermediate results) are written only once during initialization and uniformly updated [45], [54]. Based on this, GPU TEEs [45], [46], [54], [64], [71] focus on multi-granularity memory encryption solutions, generating coarse-grained metadata for low-frequency writes or large data blocks. This minimizes the height of the integrity tree and reduces storage pressure. Overall, we recommend TEE designers to design proper-grained security metadata based on accelerator access patterns.



(a) LLM startup stage and interaction overhead (initialization).



(b) LLM decoding stage and interaction overhead (communication).

Figure 4: Performance of LLM inference with S_{ME} solutions.

IME3: Inconsistent encryption granularity between CPU and accelerator incurs performance overhead. Incompatible memory protection granularity between the CPU Memory Encryption Engine (MEE) and the accelerator MEE introduces additional interaction processes and generates performance overhead. As shown in Figure 3(c), this overhead is composed of two parts: (1) the secure initialization, where the CPU and accelerator need to synchronize metadata via a trusted channel (e.g., the Security Protocol and Data Model [92]); and (2) the secure communication, where the CPU and accelerator need to negotiate protection granularity and transfer data through bounce buffers [34], [58]. To further analyze the overhead of these two parts, we conducted an evaluation using large language models (LLMs). Following state-of-the-art NPU TEEs [48], [51], [53], [66], [76], [81] based on the cycle-accurate NPU simulator SCALE-Sim [110], we set both AES and MAC latencies to 40 cycles and measured 9 LLMs (with parameter sizes ranging from 1.1 billion to 9 billion). As shown in Figure 4(a), during the secure initialization phase (before generating the first token), the CPU generates and synchronizes secure metadata based on model parameters and input, incurring 47.88% to 73.45% overhead. Additionally, in the secure communication phase, the CPU and accelerator must continuously update and verify metadata to decode subsequent tokens, resulting in 40.36% to 44.94% overhead (Figure 4(b)). We observe that the overhead of secure communication is stable and relatively smaller than that of secure initialization. This is because the decoding phase — during which numerous tokens are computed — dominates the overall inference execution time. Currently, some TDISP-based solutions [55]–[57] reduce this overhead by eliminating bounce buffers. However, when combined with accelerator memory encryption of different granularities, accelerator TEEs still suffer from the overhead of re-encryption [45], [48], [51], [53], [81] process. Thus, when designing accelerator MEEs for TEEs, developers should carefully ensure the consistency of CPU and accelerator MEE to reduce interaction overhead.

VII. Attestation

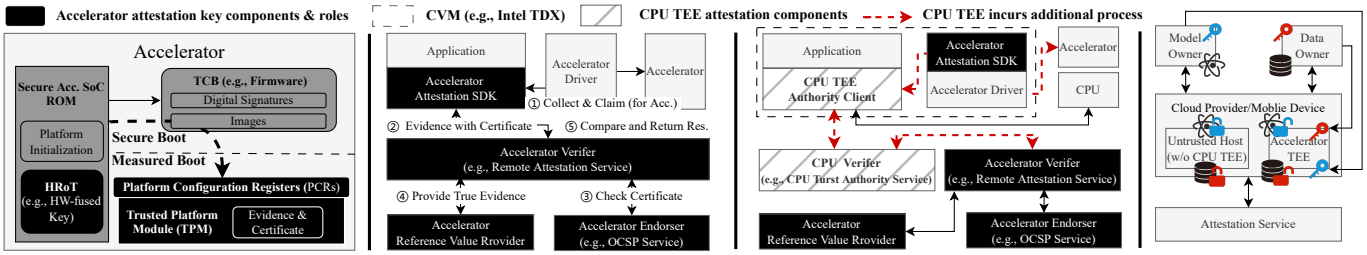
A. Solutions for Attestation

S_{AT1}: CPU HRoT-based attestation. Studies [68], [80] can reuse the CPU-side Trusted Platform Module (TPM) [111] to verify the authenticity of accelerator software stacks and the confidential accelerator tasks. These software stacks and tasks contain checksum, Hash-based Message Authentication Code (HMAC), and other signature certificates. By checking with TPM-recorded results, this solution defends against unauthorized or maliciously replaced tasks (A_{A1}). Moreover, several studies [31], [68] can leverage this solution to defend against the emulated or incorrect computing environment (A_{A2}) by attesting its execution environment or accelerator hardware.

S_{AT2}: SW-based attestation. Besides attestation based on CPU HRoT, study [65] proposes a software-based mechanism for attesting tasks and accelerators. This solution submits an attestation kernel to the accelerator to check the task execution. Software-based authentication defends against compromising the authenticity of tasks (A_{A1}) without requiring hardware support. However, it is only effective during the runtime phase and cannot establish a root trust chain. For security guarantee, the key storage and protocols negotiation can rely on the CPU-side CVM/enclaves (e.g., Intel TDX and SGX).

S_{AT3}: Accelerator HRoT-based attestation. Studies assume or equip an accelerator HRoT, such as the ROM-stored key in NVIDIA H100 [58] or the PUF-generated immutable key in FPGA [35]. Based on this, the accelerator can verify its own software in the secure boot stage, following a pre-defined chain of trust. Same as $S_{AT1,2}$, these software stacks must contain checksum and signature certificates for verification. This solution can effectively defend against adversaries who tamper with the accelerator boot images and firmware (A_{A2}).

S_{AT4}: Attestation supported by extension hardware. Studies use the HRoT in external security hardware (e.g., additional security hardware [52], [59] and the security controller [28] on the IO bus) to attest the accelerator device. This solution often requires to develop proprietary HRoT or embed third-party TPM chips (e.g, Infineon [112]) to build a root trust chain.



(a) Establishing the chain of trust based on HRoT (initialization phase)

(b) Accelerator remote attestation roles and messages (runtime phase)

(c) CPU-Accelerator integrated attestation roles and messages (e.g., Intel TDX + NVIDIA H100)

(d) Task attestation (e.g., Huawei GuardAIn)

Figure 5: The mainstream attestation process in accelerator TEEs.

Same as S_{AT3}, this solution provides attestation capability on accelerator devices and defends against A_{A2}.

B. Insights on Attestation

IAT1: Security implications of lacking attestation implementation. Accelerator TEEs require attestation to ensure the authenticity of the accelerator environment. Based on the TCG attestation protocol [113], we propose an attestation process for accelerators (shown in Figure 5). In the initialization phase, evidence (e.g., hash values and boot logs) and certificates of the accelerator environment (including firmware, kernel, and application) are collected via the HRoT and TPM (see Figure 5(a)). In the runtime phase, the Endorser and Reference Value Provider supply authentic certificates and evidence, respectively, to the Verifier (e.g., NRAS, NVIDIA Remote Authentication Service [58]) for verification (see Figure 5(b)).

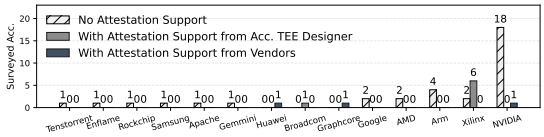


Figure 6: Attestation support in surveyed accelerators. Note that we provide detailed data in Table VIII.

Table VIII: Surveyed accelerators in Figure 6.

Acc. Vendors	Acc. Device Name (used Acc. TEE)
Tenstorrent	N150 NPU ([78])
Enflame	S60 GPU ([78])
Rockchip	RK3588S NPU ([77])
Samsung	Exynos 990 NPU* ([51], [53], [81])
Apache	VTA NPU* ([47])
Gemmini	Gemmini NPU ([37], [69])
Huawei	Ascend 910A NPU ([29], [30])
Broadcom	VideoCore IV GPU ([63])
Graphcore	GC200 IPU ([62])
Google	TPU-v1* ([48], [51], [66], [81]), TPU-v3* ([76])
AMD	Radeon RX VEGA 64 GPU ([43]), RX6900XT GPU ([61])
Arm	Mali-G71 GPU ([49], [60]), Mali-T624 GPU ([31], [68]), Mali-G610 GPU ([80]), Ethos-N77 NPU* ([53])
Xilinx	VCU118 FPGA ([59], [67]), Zynq-7000 FPGA ([42]), ZCU106 FPGA ([75]), XCZU15EG FPGA ([74]), XCZU9EG FPGA ([44]), UltraScale+ Ultra96 FPGA ([52]), Alveo U200 FPGA ([72]), ADM-PCIE-7V3 FPGA ([35])
NVIDIA	GTX 780 GPU ([34]), GTX 460 SE GPU ([67]), GTX 580 GPU ([33]), GTX 2080 GPU ([47]), GTX Titan Black GPU ([28], [34]), Tesla P40 GPU ([28]), Tesla V100 GPU ([28]), T4 GPU ([78]), L20 GPU ([78]), RTX 4090Ti GPU ([78]), RTX 3080 GPU ([82]), RTX 2080 GPU ([49]), TITAN X Pascal GPU* ([45]), Volta Arch. GPU* ([46], [64], [71]), Turning Arch. GPU* ([54]), H100 GPU ([58], [79]), A100 GPU ([65], [78]), NVDLA ([73]), Jetson AGX Orin ([32])

*Accelerator TEE uses simulators to emulate the corresponding commercial accelerators (e.g.,MGX [51] use SCALE-Sim [110] to simulate Google TPU-v1 and Samsung Exynos 990 NPU).

However, our analysis reveals that only three accelerator vendors (Huawei [29], Graphcore [62], and NVIDIA [58]) have deployed complete attestation mechanisms on their specific accelerators. Additionally, a small number of programmable accelerators (e.g., FPGAs) or platforms implement basic attestation components [35], [42], [52], [63], [72], [74], [75], while most accelerators (34 out of 44 accelerator devices used in accelerator TEEs) lack any attestation implementation (shown in Figure 6), resulting in severe authenticity vulnerabilities. Table IX shows our analysis on missing attestation components. If an accelerator TEE lacks an HRoT or Endorser, attackers can emulate or replace system components with unauthorized versions to compromise CPU TEEs. Furthermore, missing TPMs or Reference Value Providers enable attackers to bypass secure boot and inject malicious code into the accelerator TEE. For TEEs without accelerator driver protection [31], [63], [68], a recent attack [114] demonstrates that adversaries can inject malicious microcontroller unit (MCU) firmware to steal or tamper with sensitive data protected by the accelerator TEE. To address these missing security components, vendors such as AMD plan to integrate open-source Roots of Trust (e.g., Caliptra [115]) into their future GPU products. We recommend that more accelerator vendors provide robust software and hardware support for attestation.

Table IX: Security implications of missing attestation.

Components&Roles	Sec. Properties	Properties Violation → Consequence	W/O Implementation Studies Examples
HRoT&Endoser	SW&HW from authorized sources	Emulated or replaced components (e.g., injecting malicious bitstream) → Providing malicious TEE	[45], [46], [48] [49]–[51] [32], [54], [60] [64], [66], [70] [37], [71], [73] [76], [77], [81]
TPM with PCRs& Ref. Value Provider	Trusted parts must boot before untrusted parts	Violate boot order → Secure configuration bypass	
	Trusted image must be checked	Violate integrity check of boot image → Code injection in TEE	

IAT2: Potential threats of CPU-accelerator integration attestation. Most accelerator TEEs rely on CPU TEEs (see Table I) but lack a standardized attestation process for CPU-accelerator interactions. Based on the attestation mechanisms of Intel TDX and NVIDIA H100 [116], we propose a generic CPU-accelerator integrated attestation workflow (see Figure 5(c)). Compared to traditional accelerator attestation workflows, the CPU's Authority Client and Trust Authority Service act as intermediaries to relay requests and receive evidence and attestation results. This structure requires users, accelerator manufacturers, and cloud service providers to place full trust in the CPU manufacturer. However, the centralized CPU authority service may access sensitive accelerator-related information — such as accelerator identifiers, TCB details, and security configuration policies — threatening accelerator

environment and leaking sensitive data [117]. To address this, we recommend that accelerator TEEs either decouple accelerator and CPU attestation processes or integrate privacy-preserving techniques (e.g., zero-knowledge proofs, secure multi-party computation) to safeguard sensitive information.

IAT3: Inadequacy of command-level task attestation for AI tasks. Task attestation ensures the integrity of task execution on accelerators. However, current implementations (e.g., Graviton [34]) primarily focus on command-level attestation, where individual kernel operations (e.g., a memory copy in CUDA) are protected through authenticated encryption and MAC-based integrity verification. This approach fails to account for the sequential dependencies inherent in complex AI tasks, leaving them vulnerable to integrity breaches. AI models typically consist of multiple layers with interdependent operations (e.g., ReLU → Matrix Multiplication → SoftMax). In such cases, a malicious host can exploit the lack of sequence verification to inject, modify, or reorder tasks. For example, an attacker might redirect tasks to malicious binaries or leak sensitive model parameters, compromising both model confidentiality and integrity. Furthermore, AI tasks often involve multiple distrusting parties (e.g., data providers and model providers), introducing collusion risks, such as a data owner colluding with a cloud provider to steal intellectual property. To address these limitations, we build on state-of-the-art research [29] and propose an AI task attestation workflow (see Figure 5(c)). This workflow assumes each participant holds unique keys to encrypt and integrity-protect both task binaries and their execution sequences. Using the Diffie-Hellman key exchange mechanism [118], participants establish secure sessions with the attested accelerator, enabling end-to-end verification of task dependencies. Untrusted third-party platforms (e.g., cloud providers) only handle encrypted data and models. Given the increasing complexity of future AI tasks, TEE designers should adapt accelerator task attestation solutions to integrate task-specific features.

Answer to RQ2. Based on our analysis in §IV, §V, §VI, and §VII, TEE designers should account for complex software/hardware attack vectors and adopt three mainstream defense mechanisms: access control, memory encryption, and attestation. To implement access control, designers should first consider TEE deployment scenarios and limitations related to high-privilege protection, then address these challenges through hybrid solutions. For memory encryption, while TEE designers may optionally implement it to mitigate specific threats, they must factor in the significant performance overhead involved. Lastly, we recommend that TEE designers carefully evaluate attestation mechanisms and collaborate with accelerator manufacturers to ensure the authenticity.

VIII. TCB Analysis

Similar to CPU TEEs, a key factor that influences accelerator TEE's real-world deployment is the bloated trusted computing base (TCB). In this aspect, we analyze the TCB in both guests' CVM/enclaves and the system components.

Table X: System TCB size of accelerator TEEs. The red number indicates native TCB size and the green number indicates added TCB in each study. In this table, we set the adversary to access the host OS, hypervisor and general CVM/enclaves, excluding them from the system TCB.

Acc. TEE	High privilege SW		Low privilege SW (CVM/Enclave) or Sec. HW (Acc./Board/IO)
	TSM (Hyp.-level)	Firmware (Mon.-level)	
Graviton [34]	-	SGX-FW	Cmd Processor(Acc.)
HIX [33]	-	SGX-FW	IO Filter(IO)
HETEE [28]	-	-	Sec. Controller(IO)
TrustOre [42]	-	SGX-FW	TrustMod(Acc.)
Telekine [43]	-	SGX-FW	Cmd Processor(Acc.)
Ambassy [44]	-	Mon(0.5M)	Acc. Controller(Acc.)
CommonCounters [45]	-	SGX-FW	Cmd Processor(Acc.)
CURE [36]	-	Sec. Monitor(0.5K) Crypt. Op. (2.6K)	IO Filter(IO)
PSSM [46]	-	SGX-FW	Cmd Processor(Acc.)
SGX-FPGA [35]	-	SGX-FW	FPGA Sec. Monitor(Acc.)
Cronus [47]	S-Hyp(35K) mEnclave Mng(4.3K) HAL core(2.1K)	Mon(0.5M)	-
GuardNN [48]	-	-	Micro-Controller(Acc.)
LEAP [49]	-	Mon(0.5M)+0.5K	OP-TEE(0.3M)+0.7K
LITE [50]	SVSM(5K)	SEV-FW	Acc. Controller(Acc.)
MGX [51]	-	SGX-FW	Cmd Processor(Acc.)
ShEF [52]	-	-	Shield(Board)
StrongBox [31]	-	Mon(0.5M) Crypt. Op.(0.5K) Integrity Check(0.2K) Access Control(0.3K) Other Config.(0.2K)	-
TNPU [53]	-	SGX-FW	Memory Controller(Acc.)
SHM [54]	-	SGX-FW	Cmd Processor(Acc.)
Arm RME-DA [55]	RMM(33K)	Mon(0.5M)	DSM(Acc.)
AMD SEV-TIO [56]	SVSM(5K)	SEV-FW	DSM(Acc.)
Intel TDX Connect [57]	TDX Mod.(35K)	TDX-FW	DSM(Acc.)
NVIDIA H100 [58]	TDX Mod.(35K)	TDX-FW	Acc. Controller(Acc.)
AccShield [59]	TDX Mod.(35K)	TDX-FW	Sec. Mng(Board)
AvaGPU [32]	S-Hyp(35K) S2 Trans.(0.4K) Sec. GPU Mng(4.9K) Mediator(0.2K) Replayer(0.3K) Scheduler(1.6K)	Mon(0.5M)	-
GR-T [60]	-	Mon(0.5M)	-
Honeycomb [61]	SVSM(9.8K) SM, Sandbox VM(9.4K) Validator(12.3K)	SEV-FW	-
ITX [62]	-	-	ICU, CCU(Board)
MyTEE [63]	S-Hyp(35K)+1.5K	Mon(0.5M)+2.0K	-
Plutus [64]	-	-	Memory Controller(Acc.)
SAGE [65]	-	SGX-FW	Kernel Caller(Acc.)
Securator [66]	-	-	Sec. Module(Acc.)
ACAI [67]	RMM(33K)+0.4K	Mon(0.5M)+1.6K	-
CAGE [68]	RMM(33K)	Mon(0.5M) Task Mng(0.7K) Env. Protection(0.4K) GPT Opti.(0.1K) Other Config.(0.1K)	-
Dhar et al. [30]	SVSM(5K)	SEV-FW	Security Controller(IO)
HyperTEE [69]	-	Cust. M-Mon	EMS(Board)
Na et al. [70]	-	SGX-FW	Command Processor(Acc.)
Salus-FPGA [72]	-	SGX-FW	SM-Controller(Acc.)
Salus-GPU [71]	-	-	Memory Controller(Acc.)
sIOPMP [73]	-	M-Mon	IO Filter(IO)
sNPU [37]	-	Crypt. Op.(10.8K) Allocator(1.7K) Other Config.(0.1K)	Isolator, Guard(Acc.)
SrcTEE [74]	-	Mon(0.5M)	Config. Sec. Unit(Board)
T-Edge [75]	-	Mon(0.5M)	Acc. Controller(Acc.)
TensorTEE [76]	-	SGX-FW	Memory Controller(Acc.)
ASGARD [77]	S-Hyp(35K)+2.0K	Mon(0.5M)	-
ccAI [78]	TDX Mod(35K)	TDX-FW	PCIe-SC(IO)
GuardAIn [29]	-	-	Task Scheduler(Acc.)
PipeLLM [79]	TDX Mod.(35K)	TDX-FW	Acc. Controller(Acc.)
Portal [80]	RMM(33K)+0.8K	Mon(0.5M)+0.1K	CVM(26M) System Realm(0.6K) Realm VM(0.2K)
SeDA [81]	-	-	Memory Controller(Acc.)
XpuTEE [82]	VMX root+6.0K	-	-

Table XI: Codes for confidential computing in H100 driver.

	Function	LoC
Security	Access Control	2547
	Memory Encryption	2456
	Attestation	3070
Functionality	Fault/Interrupt Handler	263
	Runtime Utility	487
	CC Flags	789
Total		9612

ITC1: Large TCB additions and changes for confidentiality support. Most commercial accelerators are not specially designed for accelerator TEEs. However, for accelerator vendors, introducing TEE supports in non-confidential accelerators can introduce large TCB additions and changes to the accelerator driver. For this problem, we analyze the source code of NVIDIA's open-source driver (which serves both H100 GPU and general GPUs). Specifically, we measure the source code related to TEE support (e.g., confidential access control flags such as `CC`, encryption such as `AES`, and attestation such as `SPDM`) with widely-used tool *cloc* [119]. Table XI shows our results. The official NVIDIA driver introduces 2.5K LoC on access control, 2.4K LoC on memory encryption, and 3.0K LoC on attestation. Besides the major security support, NVIDIA also introduces 1.5K LoC for essential functionality support (e.g., handling faults/exceptions, managing `CC` runtime, and indicating `CC` status). Thus, TEE support on general accelerators can require non-negligible additions/changes, finally converting to the large TCB in guest CVM/enclaves.

ITC2: Large TCB from supporting varied accelerator software. Leveraging TEE to secure accelerator software (S_{AC2}) is a naive method to protect accelerator workloads/environments [33], [47], [58], [67]. However, such a solution can add non-negligible TCB to CVMs/enclaves and thus influence their security. As shown in Table XII, the widely used NVIDIA and AMD GPUs introduce 1.4M LoC and 5.0M LoC to CVM/enclave to protect their kernel-layer drivers, respectively. For NVIDIA GPUs, although their official compiler (i.e., CUDA) is closed-source, one of the non-official implementations, gdev [120], requires 0.3M LoC TCB addition. Xilinx FPGAs also provide a standardized runtime (i.e., XRT [121]) with 0.3M LoC. Worse, with accelerator hardware updates, the corresponding software stacks also increase their size to support more accelerator functions. For instance, when Arm upgrades its Mali GPUs to the third generation (e.g., Mali G71 GPUs), the size of the GPU driver also extends from 47K to 0.1M LoC. Considering most platforms support more than one type of accelerators, the guests' CVM/enclaves can require more TCB to support these accelerators.

ITC3: Abusing TCB addition in high-privilege software. Accelerator TEEs may abuse TCB additions in high-privilege software (i.e., TSM and firmware), potentially undermining CPU-side system security. As shown in Table X, Cronus [47] and Honeycomb [61] introduce 6.4K and 22K lines of code (LoC) additions, respectively, to lightweight TSMs (e.g., Arm Hafnium [137] and AMD SVSM [138])—largely expanding the TCB of this component. For firmware-based protection (S_{AC4}), four studies [31], [63], [67], [68] add >1.0K LoC to the monitor — a thin TCB but in the highest-privilege component. These TCB additions can thus introduce a non-negligible attack surface to the CPU-side system. However, such additions can be decoupled into low-privilege components. For instance, Cronus [47] and CAGE [68] leverage hundreds of LoCs to manage accelerator TEE isolation (e.g., a 4.3K enclave manager in Cronus and 0.5K GPC-related configuration in CAGE). If these codes are decoupled to a CVM (e.g., Portal [80]), a co-processor unit, or an external peripheral (e.g., HETEE [28]), TCB additions would be effectively reduced without interfering with the TSM/Monitor's native functionality and security. Overall, TEE designers should flexibly invoke low-privilege components to minimize TCB additions in high-privilege software.

IX. Compatibility Discussions

For compatibility issues, we categorize them into two significant aspects: (1) *multi-type* issues, which indicate the compatibility problems on supporting multiple types of hardware platforms, which can equip varied CPU, TEE, and accelerator; and (2) *plug-and-play* issues, which indicate the compatibility problems on supporting native user-level software and native platform hardware. We summarize the compatibility requirements of accelerator TEEs in Table XIII and detail the compatibility issues as follows.

ICP1: Lack of multi-type CPU TEE support. Deploying an accelerator TEE across different platforms presents significant challenges, as many TEEs are designed for a single architecture or specific type. Most accelerator TEEs face this compatibility issue. On one hand, some TEEs implement access control using unique security mechanisms (such as Arm GPC in CAGE [68] and Portal [80]) and architecture-specific privileges (e.g., the VMPL0 layer in Honeycomb [61]). Consequently, they cannot be supported on other CPU architecture platforms with different security primitives. On the other hand, accelerator TEEs may support only application-layer enclaves [33], [35], kernel-layer CVMs [67], or a unified secure world [31]. These variations result in substantial differences in supported software and execution workflows, requiring non-trivial effort to redesign TEE management systems. To address this, we recommend that TEE designers adopt a generalized framework encompassing access control, memory encryption, and attestation to enhance cross-platform compatibility.

ICP2: Lack of multi-type accelerator support. A large number of accelerator TEEs (43/51 studies) focus on a single type of accelerator device, such as GPUs [33], [34], [68], NPUs [37], [53], TPUs [59], or FPGAs [35], [72], [75]. Although accelerators share similar interaction methods with CPUs (i.e., DMA and MMIO), their task execution workflows, supported libraries/drivers, and hardware configurations (e.g., MMIO and DMA settings) can vary significantly. For example, accelerator drivers may feature unique memory management (e.g., unified virtual memory in NVIDIA drivers) or other specialized functions. Several user-layer libraries (e.g., CUDA [98] and OpenCL [99]) are even closed-source. These factors introduce non-negligible modifications when

Table XII: Size of accelerator software stacks.

Software Stack		Supported Acc.	TCB Size (ver. of src.)	Acc. TEE
NVIDIA GPU	official CUDA toolkit [98]	NVIDIA GPU [14]	N/A[2]	[28], [32], [50], [58], [65], [78], [79], [82]
	official kernel driver [122]	NVIDIA GPU [14]	1.4M (v575.64.05 [122])	[28], [50], [58], [78], [79], [82]
	gdev [120]	NVIDIA GPU [14]	0.3M (latest [123])	[33], [34], [47], [67]
	nouveau [124]	NVIDIA GPU [14]	0.1M (in Linux v6.16 [125])	[34], [47], [67]
AMD GPU	ROCm [126]	AMD Radeon GPU [15]	10M (latest [126])[1]	[43], [61]
	AMD GPU driver	AMD Radeon GPU [15]	5.0M (in Linux v6.16 [125])	[43], [61]
Arm Mali GPU	Bifrost driver [89]	Mali G71/G610 GPU [16]	0.1M (r54p1 [89])	[49], [60], [80]
	Midgard driver [127]	Mali T6xx/T7xx/T8xx GPU [16]	47K (r32p0 [127])	[31], [68]
OpenCL [99]		Mali GPU [16]	N/A[2]	[31], [49], [60], [68], [80]
Xilinx FPGA	Xilinx DMA drivers [128]	Xilinx FPGA [24]	7.6K (latest [128])	[67]
	XRT [121]	Xilinx FPGA [24]	0.3M (v2.19.194 [121])	[52]
Coyote FPGA software stack [129]		Xilinx FPGA [24]	7.5K (v0.2.1 [130])	[59]
Huawei NPU	Ascend software [131]	Huawei Ascend NPU [21]	N/A[2]	[29], [30]
VTA NPU	vta-driver [132]	VTA NPU [133]	2.7K (latest [132])	[47]
Arm NPU	Ethos-N driver stack [134]	Arm Ethos-N77 NPU [19]	72K (v25.03 [134])	[53]
Samsung NPU	Exynos driver [135]	Samsung Exynos NPU [20]	17K (latest [135])	[53]
NVIDIA DNN Acc.	NVDLA software [136]	NVDLA DNN Acc. [27]	0.4M (v1.2.0 [136])	[37], [73]

[1] The ROCm is a complex software stacks. In this paper we mainly measure the TCB of entire compilers (e.g., amd-llvm and HIP), ROCR and ROCm runtime.
[2] Closed-source software.

Table XIII: Compatibility requirements of accelerator TEEs. ●; ○: Satisfied; Not satisfied. **MT TEE**: Multi-type TEE architecture support. **MT Acc.**: Multi-type accelerator support. **PP SW**: Plug-and-play support on accelerator software. **PP Plat**: Plug-and-play support on accelerator-equipped platform.

Acc. TEE	MT TEE	MT Acc.	PP SW	PP Plat.
Graviton [34]	○	○	○	○
HIX [33]	○	○	○	○
HETEE [28]	●	●	○	●
TrustOre [42]	○	○	○	○
Telekine [43]	○	○	○	○
Ambassy [44]	○	○	●	○
CommonCounters [45]	○	○	○	○
CURE [36]	●	○	●	○
PSSM [46]	○	○	○	○
SGX-FPGA [35]	○	○	○	○
Cronus [47]	○	●	●	●
GuardNN [48]	●	○	●	○
LEAP [49]	○	○	●	●
LITE [50]	○	○	○	○
MGX [51]	○	○	○	○
ShEF [52]	●	○	○	●
StrongBox [31]	○	○	○	●
TNPU [53]	○	○	○	○
SHM [54]	○	○	○	○
Arm RME-DA [55]	○	●	○	○
AMD SEV-TIO [56]	○	●	○	○
Intel TDX Connect [57]	○	●	○	○
NVIDIA H100 [58]	○	○	○	○
AccShield [59]	○	○	●	●
AvaGPU [32]	○	○	○	●
GR-T [60]	○	○	○	●
Honeycomb [61]	○	○	○	●
ITX [62]	●	○	○	○
MyTEE [63]	○	○	○	●
Plutus [64]	●	○	●	○
SAGE [65]	○	○	○	●
Securator [66]	●	○	●	○
ACAI [67]	○	●	●	○
CAGE [68]	○	○	○	●
Dhar et al. [30]	●	●	○	●
HyperTEE [69]	●	○	●	○
Na et al. [70]	●	○	○	○
Salus-FPGA [72]	○	○	○	○
Salus-GPU [71]	●	○	●	○
sIOPMP [73]	○	○	●	○
sNPU [37]	○	○	●	○
SrcTEE [74]	○	○	●	○
T-Edge [75]	○	○	●	○
TensorTEE [76]	○	○	○	○
ASGARD [77]	○	○	●	●
ccAI [78]	●	●	●	●
GuardAIn [29]	●	○	○	○
PipeLLM [79]	○	○	○	○
Portal [80]	○	○	○	●
SeDA [81]	●	○	●	○
XpuTEE [82]	○	○	○	●

integrating native protection mechanisms with new accelerator software and hardware. Thus, TEE designers should prioritize a standardized CPU-accelerator interaction framework.

ICP3: Non-trivial changes on accelerator software. Accelerator TEEs may require substantial modifications to native accelerator drivers, libraries, and applications, driven by two key considerations. First, accelerator TEEs may add or modify native driver functions to support TEE security mechanisms. Such changes include managing buffers and page tables [31], [68], collecting task information [28], controlling secure accelerator execution [33], [34], interacting with new security modules in CVMs/enclaves [32], and other functional adjustments. These modifications also necessitate accelerator applications to invoke TEE-related APIs [28], [31]. Second, some accelerator TEEs [47], [61] may further decouple complex software stacks for security concerns. This is because most accelerator drivers and libraries are relatively large and contain buggy code, which poses risks to the security of CVM/enclaves.

ICP4: Non-trivial changes on platform hardware. Finally, accelerator TEEs may involve modifications to accelerator hardware, the PCIe I/O bus, or even CPU instruction sets. Such changes are common in *Acc.-type* and *Mix-type* accelerator TEEs [34], [37], [59], [62], and are partially adopted in some *Host-type* studies [33], [36]. Given the hardware variability across different CPUs and accelerators, reproducing these accelerator TEEs on new platforms requires addressing non-trivial compatibility challenges. For cloud providers leveraging commercial accelerator TEEs, we strongly recommend avoiding hardware modifications in accelerator TEE design.

Answer to RQ3. Based on our analysis in §VIII and §IX, TEE designers should carefully address the TCB and compatibility limitations associated with deploying accelerator TEEs. In terms of TCB, designers should consider the bloated size of accelerator software stacks and the TCB expansion resulting from high-privilege components. For compatibility, designers should maximize compliance with the multi-type and plug-and-play compatibility requirements for accelerator software, hardware, and the platforms they are deployed on.

X. Other Discussions

Availability attacks. Defending against availability attacks (e.g., Denial-of-Service) is generally regarded as orthogonal work in most accelerator TEEs [45], [47], [50], [53], [58], [61], [65]. Currently, few accelerator TEEs (e.g., AvaGPU [32]) systematically analyzes availability threats related to GPU task execution delays and preemption, addressing them through a CPU-GPU joint scheduling framework. Nevertheless, solutions from other CPU TEEs or non-accelerator I/O TEEs can be migrated to accelerator TEEs to mitigate availability threats. For example, Hora [139] designs a formally verified scheduler to ensure periodic availability during application execution, while Aion [140] implements an enclave-side scheduler to guarantee resource fairness for protected applications.

Other security requirements on deployment. As highlighted earlier in **IAC1**, **IME1**, and **IAT1**, we analyzed cloud and

endpoint deployment challenges from the perspectives of attack vectors and design preferences. However, TEE designers must also account for other real-world considerations. For example, from a user requirement perspective, cloud accelerator TEEs typically serve large enterprises with stringent confidentiality needs. These enterprises may demand rigorous attestation processes for third-party accelerator firmware and remote attestation verifier. In contrast, endpoint accelerator TEEs generally cater to personal users who do not require such strict measures. These deployment-specific factors partially guide TEE designers in defining their security requirements.

TDISP. The industry has proposed an accelerator TEE framework, called the TEE Device Interface Security Protocol (TDISP) [86]. This framework provides a formal system overview and protocols (e.g., SPDM [92]) for CPU-side security components, PCIe, and accelerator devices. Currently, manufacturers of PCIe, CPUs, and accelerators have begun supporting TDISP, including through the Compute Express Link (CXL) IDE specification [141], Intel TDX Connect [57], and NVIDIA B200 GPUs [142]. Several studies focus on TDISP-based systems (e.g., ACAI [67]) or conduct related security verification for accelerator and CPU hardware [143]–[146]. However, TDISP is not a definitive solution for building accelerator TEEs. Currently, TDISP-compliant CPUs and accelerators have not adequately addressed key blind spots, such as communication-related memory encryption overhead (**IME2**), threats within the CPU-accelerator trust chain (**IAT1**), the introduction of large TCBs into CVMs (**ITC1**), and poor compatibility with general accelerators/platforms (**ICP4**).

XI. Related Works

Related survey works. Current surveys have yet to propose a detailed analysis for accelerator TEE designs. Thus, they fail to answer our three research questions. Several studies [39]–[41], [147], [148] survey TEE threats and protection on varied CPU architectures, while they lack the same analysis on the accelerator side. For accelerator surveys, most surveys [149]–[152] on accelerator security focus on threats. Wang [153] discusses several threats and defense mechanisms on GPU TEEs, while they have yet to consider TEE for other accelerators (e.g., NPU TEE). Compared to these studies, this paper analyzes the detailed attack vectors and accelerator TEE framework, systematically analyzing defense mechanisms and their security/TCB/compatibility insights.

Non-accelerator I/O TEEs. Several studies extend TEE design to protect I/O operations [154]–[156], adopting security solutions similar to those used in accelerator TEEs. For example, Keystone [155] configures RISC-V's M-mode monitor (similar to S_{AC4}) and leverages the RISC-V PMP technique to secure CPU-device communication. SGXIO [154] secures I/O interactions through SGX-like enclaves (similar to S_{AC2}) and a trusted hypervisor (similar to S_{AC3}). While these TEEs have the potential to support accelerator computing, they lack detailed mechanisms for managing accelerators.

Additionally, some I/O TEEs are directly implemented on FPGAs, leveraging the hardware's high programmability to implement or emulate CPU-side security primitives (e.g., RISC-V PMP). Many FPGAs (e.g., Xilinx FPGAs [24]) also integrate a TrustZone-supported Arm CPU core. These features enable I/O TEEs (e.g., HECTOR-V [157], Notary [156], Split-Trust [158], and Iso-X [159]) to verify secure I/O communication. However, these solutions do not implement onboard accelerator units (e.g., a DNN IP core) and lack detailed protections for the accelerator's workloads and computing environments. For this reason, we do not analyze these studies.

Other accelerator studies. Numerous accelerator-focused studies have identified orthogonal blind spots in accelerator-equipped systems. For example, Pichai et al. [160] focus on design flaws in GPU-side access control (i.e., rebuilding the CPU MMU on GPUs) and its performance overhead. Other works discuss performance challenges related to accelerator Translation Lookaside Buffers (TLBs) [161]–[163], cache management [164], [165], scheduling [166], and system calls [167]. These studies do not focus on security issues in accelerator TEE development and typically study a specific accelerator type (e.g., GPUs). However, we believe their insights and solutions can potentially be adapted to accelerator TEEs, enhancing TEE performance and other key aspects.

XII. Conclusion

Accelerator TEE is a rising technique to protect sensitive task computing on accelerators. Although the industry and academy propose varied accelerator TEE designs, studies have yet to provide a detailed comparison of these designs and analyze the pros and cons of their security solutions. In this paper, we provide a systematical analysis of accelerator TEE studies. We categorize the attack vectors and accelerator TEE designs. Based on this, we analyze three mainstream security mechanisms: access control, memory encryption, and attestation, with summarizing detailed solutions and insights. Lastly, we find that most accelerator TEEs are challenging to deploy on real-world devices because of their non-negligible TCB and compatibility issues. Our analysis will provide insightful suggestions on building an accelerator TEE in the future.

Acknowledgments

We would like to thank the anonymous reviewers and COMPASS members for their insightful comments. This work is partly supported by the National Natural Science Foundation of China under Grant No.U2541211, No.62372218, No.U24A6009. This work is also supported in part by HK RGC Collaborative Research Fund (No.C5032-23GF), and Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University. This work is also in part supported by Ant Group.

References

[1] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, 2016.
[2] AMD, "AMD Secure Encrypted Virtualization (SEV)," https://developer.amd.com/sev/, 2022.
[3] ARM, "TRUSTZONE FOR CORTEX-A," https://www.arm.com/technologies/trustzone-for-cortex-a, 2024.
[4] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A Secure Database using SGX," in *39th IEEE SP*, 2018.

[5] D. Bogatov, G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill, "εpsolute: Efficiently querying databases while providing differential privacy," in *28th ACM CCS*, 2021.

[6] T. Kim, J. Park, J. Woo, S. Jeon, and J. Huh, "ShieldStore: Shielded In-memory Key-value Storage with SGX," in *14th EuroSys*, 2019.

[7] D. Lu, M. Shi, X. Ma, X. Liu, R. Guo, T. Zheng, Y. Shen, X. Dong, and J. Ma, "Smaug: A TEE-Assisted Secured SQLite for Embedded Systems," *IEEE TDSC*, 2022.

[8] F. Mo, Z. Tarkhani, and H. Haddadi, "Machine Learning with Confidential Computing: A Systematization of Knowledge," *ACM CSUR*, 2024.

[9] Z. Zhang, C. Gong, Y. Cai, Y. Yuan, B. Liu, D. Li, Y. Guo, and X. Chen, "No Privacy Left Outside: On the (In-)Security of TEE-Shielded DNN Partition for On-Device ML," in *45th IEEE SP*, 2024.

[10] W. Xu, H. Zhu, Y. Zheng, F. Wang, J. Hua, D. Feng, and H. Li, "ToNN: An Oblivious Neural Network Prediction Scheme With Semi-Honest TEE," *IEEE TIFS*, 2024.

[11] C. Liu, H. Guo, M. Xu, S. Wang, D. Yu, J. Yu, and X. Cheng, "Extending On-chain Trust to Off-chain – Trustworthy Blockchain Data Collection using Trusted Execution Environment (TEE) ," *IEEE ToC*, 2022.

[12] S. Matetic, K. Wüst, M. Schneider, K. Kostiainen, G. Karame, and S. Capkun, "BITE: Bitcoin Lightweight Client Privacy using Trusted Execution," in *28th USENIX Security*, 2019.

[13] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: A Secure Payment Network with Asynchronous Blockchain Access," in *27th ACM SOSP*, 2019.

[14] NVIDIA Corporation, "Graphics Cards," https://www.nvidia.com/en-us/geforce/graphics-cards/, 2023.

[15] AMD, "AMD Radeon™ RX Graphics Cards," https://www.amd.com/en/graphics/radeon-rx-graphics, 2023.

[16] ARM, "Arm Mali Graphics Processing Units (GPUs)," https://developer.arm.com/ip-products/graphics-and-multimedia/mali-gpus, 2023.

[17] Qualcomm, "Adreno Graphics Processing Units," https://developer.qualcomm.com/software/adreno-gpu-sdk/gpu/, 2022.

[18] Apple, "Discover Metal enhancements for A14 Bionic," https://developer.apple.com/videos/play/tech-talks/10858/, 2022.

[19] ARM, "Ethos-N78," https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-n78, 2023.

[20] Samsung, "NPU (Neural Processing Units)," https://semiconductor.samsung.com/support/tools-resources/dictionary/the-neural-processing-unit-npu-a-brainy-next-generation-semiconductor/, 2024.

[21] H. Liao, J. Tu, J. Xia, H. Liu, X. Zhou, H. Yuan, and Y. Hu, "Ascend: a Scalable and Unified Architecture for Ubiquitous Deep Neural Network Computing: Industry Track Paper," in *27th IEEE HPCA*, 2021.

[22] Rockchip Electronics CO., LTD., "RK3588 Brief Datasheet," https://www.rock-chips.com/uploads/pdf/2022.8.26/192/RK3588%20Brief%20Datasheet.pdf, 2025.

[23] Google, "Accelerate AI development with Google Cloud TPUs," https://cloud.google.com/tpu, 2024.

[24] AMD, "FPGA Leadership Across Multiple Process Nodes," https://www.xilinx.com/products/silicon-devices/fpga.html, 2024.

[25] Intel Corporation, "Altera FPGAs and Programmable Solutions," https://www.intel.com/content/www/us/en/products/programmable.html, 2024.

[26] Lattice Semiconductor, "Avant-X High-Speed Mid-Range FPGA," https://www.latticesemi.com/Products/FPGAandCPLD/Avant-X, 2025.

[27] NVIDIA Corporation, "NVDLA," https://nvdla.org/, 2025.

[28] J. Zhu, R. Hou, X. Wang, W. Wang, J. Cao, B. Zhao, Z. Wang, Y. Zhang, J. Ying, L. Zhang *et al.*, "Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment," in *41st IEEE SP*, 2020.

[29] A. Dhar, C. Thorens, L. M. Lazier, and L. Cavigelli, "GuardAIn: Protecting Emerging Generative AI Workloads on Heterogeneous NPU," in *46th IEEE SP*, 2025.

[30] A. Dhar, S. Sridhara, S. Shinde, S. Capkun, and R. Andri, "Confidential Computing with Heterogeneous Devices at Cloud-Scale," in *40th ACSAC*, 2024.

[31] Y. Deng, C. Wang, S. Yu, S. Liu, Z. Ning, K. Leach, J. Li, S. Yan, Z. He, J. Cao *et al.*, "Strongbox: A GPU TEE on Arm Endpoints," in *29th ACM CCS*, 2022.

[32] J. Wang, Y. Wang, and Z. Ning, "Secure and Timely GPU Execution in Cyber-physical Systems," in *30th ACM CCS*, 2023.

[33] I. Jang, A. Tang, T. Kim, S. Sethumadhavan, and J. Huh, "Heterogeneous Isolated Execution for Commodity GPUs," in *24th ACM ASPLOS*, 2019.

[34] S. Volos, K. Vaswani, and R. Bruno, "Graviton: Trusted Execution Environments on GPUs," in *13th USENIX OSDI*, 2018.

[35] K. Xia, Y. Luo, X. Xu, and S. Wei, "SGX-FPGA: Trusted Execution Environment for CPU-FPGA Heterogeneous Architecture," in *58th ACM/IEEE DAC*, 2021.

[36] R. Bahmani, F. Brasser, G. Dessouky, P. Jauernig, M. Klimmek, A.-R. Sadeghi, and E. Stapf, "CURE: A Security Architecture with CUstomizable and Resilient Enclaves," in *30th USENIX Security*, 2021.

[37] E. Feng, D. Feng, D. Du, Y. Xia, and H. Chen, "sNPU: Trusted Execution Environments on Integrated NPUs," in *51st ACM/IEEE ISCA*, 2024.

[38] E. López-Morales, U. Planta, C. Rubio-Medrano, A. Abbasi, and A. A. Cardenas, "SoK: Security of Programmable Logic Controllers," in *33rd USENIX Security*, 2024.

[39] L. Zhao, H. Shuang, S. Xu, W. Huang, R. Cui, P. Bettadpur, and D. Lie, "SoK: Hardware Security Support for Trustworthy Execution," *arXiv preprint arXiv:1910.04957*, 2019.

[40] M. Schneider, R. J. Masti, S. Shinde, S. Capkun, and R. Perez, "SoK: Hardware-supported Trusted Execution Environments," *arXiv preprint arXiv:2205.12742*, 2022.

[41] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems," in *41st IEEE SP*, 2020.

[42] H. Oh, A. Ahmad, S. Park, B. Lee, and Y. Paek, "Trustore: Side-channel resistant storage for sgx using intel hybrid cpu-fpga," in *27th ACM CCS*, 2020.

[43] T. Hunt, Z. Jia, V. Miller, A. Szekely, Y. Hu, C. J. Rossbach, and E. Witchel, "Telekine: Secure Computing with Cloud GPUs," in *17th USENIX NSDI*, 2020.

[44] D. Hwang, S. Yeleuov, J. Seo, M. Chung, H. Moon, and Y. Paek, "Ambassy: A Runtime Framework to Delegate Trusted Applications in an ARM/FPGA Hybrid System," *IEEE TMC*, 2021.

[45] S. Na, S. Lee, Y. Kim, J. Park, and J. Huh, "Common Counters: Compressed Encryption Counters for Secure GPU Memory," in *27th IEEE HPCA*, 2021.

[46] S. Yuan, Y. Solihin, and H. Zhou, "PSSM: Achieving Secure Memory for GPUs with Partitioned and Sectored Security Metadata," in *35th ACM International Conference on Supercomputing*, 2021.

[47] J. Jiang, J. Qi, T. Shen, X. Chen, S. Zhao, S. Wang, L. Chen, G. Zhang, X. Luo, and H. Cui, "CRONUS: Fault-isolated, Secure and High-performance Heterogeneous Computing for Trusted Execution Environment," in *55th IEEE/ACM MICRO*, 2022.

[48] W. Hua, M. Umar, Z. Zhang, and G. E. Suh, "GuardNN: Secure Accelerator Architecture for Privacy-Preserving Deep Learning," in *59th ACM/IEEE DAC*, 2022.

[49] L. Sun, S. Wang, H. Wu, Y. Gong, F. Xu, Y. Liu, H. Han, and S. Zhong, "LEAP: TrustZone Based Developer-Friendly TEE for Intelligent Mobile Apps," *IEEE TMC*, 2022.

[50] A. W. B. Yudha, J. Meyer, S. Yuan, H. Zhou, and Y. Solihin, "LITE: A Low-Cost Practical Inter-Operable GPU TEE," in *36th ACM International Conference on Supercomputing*, 2022.

[51] W. Hua, M. Umar, Z. Zhang, and G. E. Suh, "MGX: Near-zero Overhead Memory Protection for Data-Intensive Accelerators," in *49th ACM/IEEE ISCA*, 2022.

[52] M. Zhao, M. Gao, and C. Kozyrakis, "ShEF: Shielded Enclaves for Cloud FPGAs," in *27th ACM ASPLOS*, 2022.

[53] S. Lee, J. Kim, S. Na, J. Park, and J. Huh, "TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit," in *28th IEEE HPCA*, 2022.

[54] S. Yuan, A. Awad, A. W. B. Yudha, Y. Solihin, and H. Zhou, "Adaptive Security Support for Heterogeneous Memory on GPUs," in *28th IEEE HPCA*, 2022.

[55] ARM, "Arm Realm Management Extension (RME) System Architecture," https://developer.arm.com/documentation/den0129/latest/, 2023.

[56] AMD, "AMD SEV-TIO: Trusted I/O for SecureEncrypted Virtualization," https://www.amd.com/content/dam/amd/en/documents/developer/sev-tio-whitepaper.pdf, 2023.

[57] Intel Corporation, "Intel TDX Connect Architecture Specification ," https://www.intel.com/content/www/us/en/content-details/773614/intel-tdx-connect-architecture-specification.html, 2023.

[58] NVIDIA Corporation, "NVIDIA CONFIDENTIAL COMPUTING," https://www.nvidia.com/en-us/data-center/solutions/confidential-computing/, 2022.

[59] W. Ren, W. Kozlowski, S. Koteshwara, M. Ye, H. Franke, and D. Chen, "Accshield: a new trusted execution environment with machine-learning accelerators," in *60th ACM/IEEE DAC*, 2023.

[60] H. Park and F. X. Lin, "Safe and Practical GPU Computation in TrustZone," in *18th EuroSys*, 2023.

[61] H. Mai, J. Zhao, C. Kozyrakis, M. Gao, H. Zheng, Q. Li, Z. Liu, C. Wang, H. Cui, and X. Feng, "Honeycomb: An Secure, Efficient GPU Execution Environment with Minimal TCB," in *17th USENIX OSDI*, 2023.

[62] K. Vaswani, S. Volos, C. Fournet, A. N. Diaz, K. Gordon, B. Vembu, S. Webster, D. Chisnall, S. Kulkarni, G. Cunningham *et al.*, "Confidential Computing within an AI Accelerator," in *2023 USENIX ATC*, 2023.

[63] S.-K. Han and J. Jang, "MyTEE: Own the Trusted Execution Environment on Embedded Devices," in *30th NDSS*, 2023.

[64] R. Abdullah, H. Zhou, and A. Awad, "Plutus: Bandwidth-Efficient Memory Security for GPUs," in *29th IEEE HPCA*, 2023.

[65] A. Ivanov, B. Rothenberger, A. Dethise, M. Canini, T. Hoefler, and A. Perrig, "SAGE: Software-based Attestation for GPU Execution," in *2023 USENIX ATC*, 2023.

[66] N. Shrivastava and S. R. Sarangi, "Securator: A Fast and Secure Neural Processing Unit," in *29th IEEE HPCA*, 2023.

[67] S. Sridhara, A. Bertschi, B. Schlüter, M. Kuhne, F. Aliberti, and S. Shinde, "ACAI: Protecting Accelerator Execution with Arm Confidential Computing Architecture," *33rd USENIX Security*, 2024.

[68] C. Wang, F. Zhang, Y. Deng, K. Leach, J. Cao, Z. Ning, S. Yan, and Z. He, "CAGE: Complementing Arm CCA with GPU Extensions," in *31st NDSS*, 2024.

[69] Y. Bai, P. Li, Y. Huang, M. C. Huang, S. Zhao, L. Zhao, F. Zhang, D. Meng, and R. Hou, "HyperTEE: A Decoupled TEE Architecture with Secure Enclave Management," in *57th IEEE/ACM MICRO*, 2024.

[70] S. Na, J. Kim, S. Lee, and J. Huh, "Supporting Secure Multi-GPU Computing with Dynamic and Batched Metadata Management," in *30th IEEE HPCA*, 2024.

[71] R. Abdullah, H. Lee, H. Zhou, and A. Awad, "Salus: Efficient Security Support for CXL-Expanded GPU Memory," in *30th IEEE HPCA*, 2024.

[72] Y. Zou, Y. Li, S. Wang, L. Su, Z. Gu, Y. Lu, Y. Guan, D. Niu, M. Gao, Y. Xie *et al.*, "Salus: A Practical Trusted Execution Environment for CPU-FPGA Heterogeneous Cloud Platforms," in *29th ACM ASPLOS*, 2024.

[73] E. Feng, D. Feng, D. Du, Y. Xia, W. Zheng, S. Zhao, and H. Chen, "sIOPMP: Scalable and Efficient I/O Protection for TEEs," in *29th ACM ASPLOS*, 2024.

[74] Y. Wang, X. Chang, H. Zhu, J. Wang, Y. Gong, and L. Li, "Towards Secure Runtime Customizable Trusted Execution Environment on FPGA-SoC," *IEEE ToC*, 2024.

[75] J. Shen, Y. Chen, W.-F. Wong, and E.-C. Chang, "T-Edge: Trusted Heterogeneous Edge Computing," *40th ACSAC*, 2024.

[76] H. Han, X. Zheng, Y. Wen, Y. Hao, E. Feng, L. Liang, J. Mu, X. Li, T. Ma, P. Jin *et al.*, "TensorTEE: Unifying Heterogeneous TEE Granularity for Efficient Secure Collaborative Tensor Computing," *29th ACM ASPLOS*, 2024.

[77] M. Moon, M. Kim, J. Jung, and D. Song, "ASGARD: Protecting On-Device Deep Neural Networks with Virtualization-Based Trusted Execution Environments," in *32nd NDSS*, 2025.

[78] C. Wang, D. Tang, C. Ci, J. Huang, Y. Xu, F. Zhang, J. Cao, J. Song, S. Yan, T. Wei *et al.*, "ccAI: A Compatible and Confidential System for AI Computing," in *58th IEEE/ACM MICRO*, 2025.

[79] Y. Tan, C. Tan, Z. Mi, and H. Chen, "PipeLLM: Fast and Confidential Large Language Model Services with Speculative Pipelined Encryption," in *30th ACM ASPLOS*, 2025.

[80] F. Sang, J. Lee, X. Zhang, and T. Kim, "PORTAL: Fast and Secure Device Access with Arm CCA for Modern Arm Mobile System-on-Chips (SoCs)," in *46th IEEE SP*, 2025.

[81] W. Xuan, Z. Wang, L. Feng, N. Lin, Z. Xuan, R. Fu, T.-Y. Ho, Y. Jiao, and L. Liang, "SeDA: Secure and Efficient DNN Accelerators with Hardware/Software Synergy," in *62nd ACM/IEEE DAC*, 2025.

[82] S. Fan, Z. Hua, Y. Xia, and H. Chen, "XpuTEE: A High-Performance and Practical Heterogeneous Trusted Execution Environment for GPUs," *ACM ToCS*, 2025.

[83] A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono, "SoK: Security and Privacy of Blockchain Interoperability," in *45th IEEE SP*, 2024.

[84] L. Zhou, X. Xiong, J. Ernstberger, S. Chaliasos, Z. Wang, Y. Wang, K. Qin, R. Wattenhofer, D. Song, and A. Gervais, "SoK: Decentralized Finance (DeFi) Attacks," in *44th IEEE SP*, 2023.

[85] ARM, "Arm Confidential Compute Architecture," https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture, 2023.

[86] PCI-SIG, "TEE Device Interface Security Protocol (TDISP)," https://pcisig.com/tee-device-interface-security-protocol-tdisp, 2022.

[87] Intel Corporation, "Intel Trust Domain Extensions," https://cdrdv2.intel.com/v1/dl/getContent/690419, 2022.

[88] ARM, "Introducing Arm Confidential Compute Architecture guide," https://developer.arm.com/documentation/den0125/latest/, 2023.

[89] ——, "Bifrost Mali 3rd Gen GPU Architecture," https://developer.arm.com/downloads/-/Bifrost%20Mali%203rd%20Gen%20GPU%20Architecture, 2025.

[90] ——, "CoreLink TrustZone Address Space Controller TZC-380 Technical Reference Manual," https://developer.arm.com/documentation/ddi0431/latest/.

[91] ——, "Granule Protection Checks," https://developer.arm.com/documentation/den0126/0100/Granule-Protection-Checks, 2023.

[92] DMTF, "Security Protocol and Data Model (SPDM) Specification," https://dmtf7.secondrisedx.com/sites/default/files/standards/documents/DSP0274_1.0.2.pdf, 2023.

[93] PCI-SIG, "Integrity and Data Encryption (IDE) ECN Deep Dive," https://pcisig.com/sites/default/files/files/PCIe%20Security%20Webinar_Aug%202020_PDF.pdf, 2020.

[94] Alibaba Group, "Alibaba Cloud," https://www.alibabacloud.com/, 2025.

[95] NVIDIA Corporation, "NVIDIA," https://www.nvidia.com.

[96] Huawei Technologies Co., Ltd., "Huawei - Building a Fully Connected, Intelligent World," https://www.huawei.com/, 2025.

[97] X. Wu, D. J. Tian, and C. H. Kim, "Building gpu tees using cpu secure enclaves with gevisor," in *2023 ACM Symposium on Cloud Computing*, 2023.

[98] NVIDIA Corporation, "CUDA Toolkit," https://developer.nvidia.com/cuda-toolkit, 2022.

[99] ARM, "OpenCL," https://developer.arm.com/tools-and-software/graphics-and-gaming/mali-drivers/user-space, 2022.

[100] ——, "Arm System Memory Management Unit Architecture Specification," https://developer.arm.com/documentation/ihi0070/latest/, 2023.

[101] ——, "PrimeCell Infrastructure AMBA 3 TrustZone Protection Controller (BP147)," https://developer.arm.com/documentation/dto0015/latest/, 2023.

[102] RISC-V International, "The RISC-V Instruction Set Manual, Volume II: Privileged Architecture," https://github.com/riscv/riscv-isa-manual/releases/download/Privv1.12/riscv-privileged-20211203.pdf, 2021.

[103] ARM, "ARM CoreLink TZC-400 TrustZone Address Space Controller Technical Reference Manual," https://developer.arm.com/documentation/ddi0504/latest/, 2014.

[104] S. Inc., "Physical Memory Protection," https://sifive.github.io/freedom-metal-docs/devguide/pmps.html, 2025.

[105] B. Rogers, S. Chhabra, M. Prvulovic, and Y. Solihin, "Using Address Independent Seed Encryption and Bonsai Merkle Trees to Make Secure Processors OS- and Performance-Friendly," in *40th IEEE/ACM MICRO*, 2007.

[106] NVIDIA Corporation, "NVIDIA GeForce GTX 1080," https://international.download.nvidia.com/geforcecom/international/pdfs/GeForce_GTX_1080_Whitepaper_FINAL.pdf, 2017.

[107] ——, "NVIDIA Turing Architecture," https://www.nvidia.com/content/dam/en-zz/Solutions/designvisualization/technologies/turing-architecture/NVIDIA-TuringArchitecture-Whitepaper.pdf, 2018.

[108] ——, "NVIDIA AMPERE GA102 GPU ARCHITECTURE," https://www.nvidia.com/content/dam/enzz/Solutions/geforce/ampere/pdf/NVIDIA-ampere-GA102-GPUArchitecture-Whitepaper-V1.pdf, 2020.

[109] P. Colp, J. Zhang, J. Gleeson, S. Suneja *et al.*, "Protecting Data on Smartphones and Tablets from Memory Attacks," in *20th ACM ASPLOS*, 2015.

[110] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software*, 2020.

[111] tpm2 software, "tpm2-tools," https://github.com/tpm2-software/tpm2-tools, 2019.

[112] OPTIGA, "Infineon Optiga-TPM on GitHub," https://github.com/Infineon/optiga-tpm, 2025.

[113] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and Implementation of a TCG-Based Integrity Measurement Architecture," in *13th USENIX Security*, 2004.

[114] H. Lu, Y. Deng, S. Mertoguno, S. Wang, and F. Zhang, "Mole: Breaking GPU TEE with GPU-Embedded MCU," in *32nd ACM CCS*, 2025.

[115] C. Alliance, "Caliptra," https://github.com/chipsalliance/Caliptra, 2025.

[116] Intel Corporation, "Seamless Attestation with Intel Trust Authority," https://www.intel.com/content/www/us/en/content-details/843439/seamless-attestation-with-intel-trust-authority.html, 2024.

[117] J. Chuang, A. Seto, N. Berrios, S. van Schaik, C. Garman, and D. Genkin, "TEE.fail: Breaking Trusted Execution Environments via DDR5 Memory Bus Interposition," in *47th IEEE SP*, 2026.

[118] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, 1976.

[119] AlDanial, "cloc," https://github.com/AlDanial/cloc, 2021.

[120] S. Kato, M. McThrow, C. Maltzahn, and S. Brandt, "Gdev: First-Class GPU Resource Management in the Operating System," in *2012 USENIX ATC*, 2012.

[121] AMD, "XRT Official 2025.1 release," https://github.com/Xilinx/XRT/releases/tag/202510.2.19.194, 2025.

[122] NVIDIA Corporation, "NVIDIA Linux Open GPU Kernel Module Source," https://github.com/NVIDIA/open-gpu-kernel-modules/releases/tag/575.64.05, 2025.

[123] shinpei0208, "Gdev: Open-Source GPGPU Runtime and Driver Software," https://github.com/shinpei0208/gdev, 2025.

[124] Nouveau, "Nouveau: Accelerated Open Source driver for NVIDIA GPUs," https://nouveau.freedesktop.org.

[125] Torvalds, "Linux v6.16," https://github.com/torvalds/linux/releases/tag/v6.16, 2025.

[126] AMD, "AMD ROCm Software," https://github.com/ROCm, 2025.

[127] ARM, "Mali Midgard GPU Kernel Drivers," https://developer.arm.com/Downloads/-/Mali%20Midgard%20GPU%20Kernel%20Drivers#, 2025.

[128] Xilinx, "Xilinx DMA IP Reference drivers," https://github.com/Xilinx/dma_ip_drivers, 2023.

[129] D. Korolija, T. Roscoe, and G. Alonso, "Do OS abstractions make sense on FPGAs?" in *14th USENIX OSDI*, 2020.

[130] fpgasystems, "Coyote v0.2.1 - centovalli," https://github.com/fpgasystems/Coyote/releases/tag/v0.2.1, 2025.

[131] Huawei Technologies Co., Ltd., "Ascend Computing Software Download," https://support.huawei.com/enterprise/en/category/ascend-computing-pid-1557196528909?submodel=software, 2025.

[132] Apache Software Foundation, "VTA Hardware Design Stack," https://github.com/apache/tvm-vta, 2025.

[133] ——, "VTA: Versatile Tensor Accelerator," https://tvm.apache.org/docs/topic/vta/index.html, 2022.

[134] ARM, "Arm Ethos-N Driver Stack 25.03," https://github.com/ARM-software/ethos-n-driver-stack/releases/tag/25.03, 2025.

[135] Samsung, "Samsung Exynos 850," https://github.com/samsungexynos850/, 2025.

[136] NVIDIA Corporation, "NVDLA 1.2.0-OC," https://github.com/nvdla/sw/releases/tag/v1.2.0-OC.

[137] ARM, "Hafnium v2.13.0," https://github.com/TF-Hafnium/hafnium/releases/tag/v2.13.0, 2025.

[138] AMD, "SVSM," https://github.com/AMDESE/linux-svsm, 2025.

[139] D. Zueck, N. Atallah, I. Do, Z. Yao, and A. A. Sani, "Hora: High Assurance Periodic Availability Guarantee for Life-Critical Applications on Smartphones," in *15th ACM SIGOPS Asia-Pacific Workshop on System*, 2024.

[140] F. Alder, J. V. Bulck, F. Piessens, and J. T. Mühlberg, "Aion: Enabling Open Systems through Strong Availability Guarantees for Enclaves," in *28th ACM CCS*, 2021.

[141] Compute Express Link, "CXL Specification," https://computeexpresslink.org/cxl-specification/, 2025.

[142] NVIDIA Corporation, "NVIDIA DGX B200," https://www.nvidia.com/en-us/data-center/dgx-b200/, 2025.

[143] Q. Tan, Y. Fisseha, S. Chen, L. Biernacki, J.-B. Jeannin, S. Malik, and T. Austin, "Security Verification of Low-Trust Architectures," in *30th ACM CCS*, 2023.

[144] B.-Y. Huang, S. Lyubomirsky, Y. Li, M. He, G. H. Smith, T. Tambe, A. Gaonkar, V. Canumalla, A. Cheung, G.-Y. Wei *et al.*, "Application-Level Validation of Accelerator Designs Using a Formal Software/Hardware Interface," *ACM ToDAES*, 2024.

[145] H. Witharana, Y. Lyu, S. Charles, and P. Mishra, "A Survey on Assertion-based Hardware Verification," *ACM CSUR*, 2022.

[146] A. Jayasena and P. Mishra, "HIVE: Scalable Hardware-Firmware Co-Verification using Scenario-based Decomposition and Automated Hint Extraction," *IEEE TCAD*, 2024.

[147] A. Paju, M. O. Javed, J. Nurmi, J. Savimäki, B. McGillion, and B. B. Brumley, "SoK: A Systematic Review of TEE Usage for Developing Trusted Applications," in *18th International Conference on Availability, Reliability and Security*, 2023.

[148] T. Lu, "A Survey on RISC-V Security: Hardware and Architecture," *arXiv preprint arXiv:2107.04175*, 2021.

[149] H. Naghibijouybari, E. M. Koruyeh, and N. B. Abu-Ghazaleh, "Microarchitectural Attacks in Heterogeneous Systems: A Survey," *ACM CSUR*, 2023.

[150] D. G. Mahmoud, V. Lenders, and M. Stojilovic, "Electrical-Level Attacks on CPUs, FPGAs, and GPUs: Survey and Implications in the Heterogeneous Era," *ACM CSUR*, 2023.

[151] A. Akram, V. Akella, S. Peisert, and J. Lowe-Power, "SoK: Limitations of Confidential Computing via TEEs for High-Performance Compute Systems," in *2022 IEEE International Symposium on Secure and Private Execution Environment Design*, 2022.

[152] S. Mittal, S. B. Abhinaya, M. Reddy, and I. Ali, "A Survey of Techniques for Improving Security of GPUs," *Journal of Hardware and Systems Security*, 2018.

[153] Q. Wang and D. Oswald, "Confidential Computing on Heterogeneous Systems: Survey and Implications," *arXiv*, 2024.

[154] S. Weiser and M. Werner, "SGXIO: Generic Trusted I/O Path for Intel SGX," in *7th ACM CODASPY*, 2017.

[155] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An Open Framework for Architecting Trusted Execution Environments," in *15th EuroSys*, 2020.

[156] A. Athalye, A. Belay, M. F. Kaashoek, R. Morris, and N. Zeldovich, "Notary: A Device for Secure Transaction Approval," in *27th ACM SOSP*, 2019.

[157] P. Nasahl, R. Schilling, M. Werner, and S. Mangard, "HECTOR-V: A Heterogeneous CPU Architecture for a Secure RISC-V Execution Environment," in *16th ACM Asia CCS*, 2021.

[158] Z. Yao, S. M. Seyed Talebi, M. Chen, A. Amiri Sani, and T. Anderson, "Minimizing a Smartphone's TCB for Security-Critical Programs with Exclusively-Used, Physically-Isolated, Statically-Partitioned Hardware," in *21st MobiSys*, 2023.

[159] D. Evtyushkin, J. Elwell, M. Ozsoy, D. Ponomarev, N. A. Ghazaleh, and R. Riley, "Iso-X: A Flexible Architecture for Hardware-Managed Isolated Execution," in *47th IEEE/ACM MICRO*, 2014.

[160] B. Pichai, L. Hsu, and A. Bhattacharjee, "Architectural support for address translation on GPUs: Designing Memory Management Units for CPU/GPUs with unified address spaces," *ACM SIGARCH Computer Architecture News*, 2014.

[161] J. Lee, J. M. Lee, Y. Oh, W. J. Song, and W. W. Ro, "SnakeByte: A TLB Design with Adaptive and Recursive Page Merging in GPUs," in *29th IEEE HPCA*, 2023.

[162] B. Li, Y. Wang, T. Wang, L. Eeckhout, J. Yang, A. Jaleel, and X. Tang, "STAR: Sub-Entry Sharing-Aware TLB for Multi-Instance GPU," in *57th IEEE/ACM MICRO*, 2024.

[163] B. Li, J. Yin, Y. Zhang, and X. Tang, "Improving Address Translation in Multi-GPUs via Sharing and Spilling aware TLB Design," in *54th IEEE/ACM MICRO*, 2021.

[164] X. Chen, L.-W. Chang, C. I. Rodrigues, J. Lv, Z. Wang, and W.-M. Hwu, "Adaptive Cache Management for Energy-efficient GPU Computing," in *47th IEEE/ACM MICRO*, 2014.

[165] B. Li, J. Sun, M. Annavaram, and N. S. Kim, "Elastic-Cache: GPU Cache Architecture for Efficient Fine-and Coarse-Grained Cache-Line Management," in *31st IEEE International Parallel and Distributed Processing Symposium*, 2017.

[166] T. Fu, Z. Yang, Z. Ye, C. Ma, Y. Han, Y. Luo, X. Wang, and Z. Wang, "A Survey on the Scheduling of DL and LLM Training Jobs in GPU Clusters," *Chinese Journal of Electronics*, 2025.

[167] J. Vesely, A. Basu, A. Bhattacharjee, G. H. Loh, M. Oskin, and S. K. Reinhardt, "Generic System Calls for GPUs," in *45th ACM/IEEE ISCA*, 2018.