2020 IEEE Symposium on Security and Privacy

Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment

Jianping Zhu†, Rui Hou$^{\dagger *}$, XiaoFeng Wang^{+*}, Wenhao Wang†, Jiangfeng Cao†, Boyan Zhao†,
Zhongpu Wang‡, Yuhui Zhang†, Jiameng Ying†, Lixin Zhang‡, Dan Meng†
†*State Key Laboratory of Information Security, Institute of Information Engineering, CAS, and School of Cyber Security, University of Chinese Academy of Sciences;* $^{+}$*Indiana University at Bloomington;* ‡*Institute of Computing Technology, CAS*
{zhujianping, hourui, wangwenhao, caojiangfeng, zhaoboyan, zhangyuhui, yingjiameng, mengdan}@iie.ac.cn,
xw7@indiana.edu, {wangzhongpu, zhanglixin}@ict.ac.cn

***Abstract*—With its huge real-world demands, large-scale confidential computing still cannot be supported by today's Trusted Execution Environment (TEE), due to the lack of scalable and effective protection of high-throughput accelerators like GPUs, FPGAs, and TPUs etc. Although attempts have been made recently to extend the CPU-like enclave to GPUs, these solutions require change to the CPU or GPU chips, may introduce new security risks due to the side-channel leaks in CPU-GPU communication and are still under the resource constraint of today's CPU TEE.**

To address these problems, we present the first Heterogeneous TEE design that can truly support large-scale compute or data intensive (CDI) computing, without any chip-level change. Our approach, called *HETEE*, is a device for centralized management of all computing units (e.g., GPUs and other accelerators) of a server rack. It is uniquely designed to work with today's data centres and clouds, leveraging modern resource pooling technologies to dynamically compartmentalize computing tasks, and enforce strong isolation and reduce TCB through hardware support. More specifically, HETEE utilizes the PCIe ExpressFabric to allocate its accelerators to the server node on the same rack for a non-sensitive CDI task, and move them back into a secure enclave in response to the demand for confidential computing. Our design runs a thin TCB stack for security management on a security controller (SC), while leaving a large set of software (e.g., AI runtime, GPU driver, etc.) to the integrated microservers that operate enclaves. An enclaves is physically isolated from others through hardware and verified by the SC at its inception. Its microserver and computing units are restored to a secure state upon termination.

We implemented HETEE on a real hardware system, and evaluated it with popular neural network inference and training tasks. Our evaluations show that HETEE can easily support the CDI tasks on the real-world scale and incurred a maximal throughput overhead of 2.17% for inference and 0.95% for training on ResNet152.

I. Introduction

The explosive growth of the data being collected and analyzed has fueled the rapid advance in data-driven technologies and applications, which have also brought data privacy to the spotlight as never before. A large spectrum of data-centric innovations today, ranging from personalized healthcare, mobile finance to social networking, are under persistent threats of data breaches, such as Facebook data exposure [1], [2], and the growing pressure for compliance with emerging privacy laws and regulations, like GDPR (general data protection regulation) and the CCPA (California Consumer Privacy Act). As a result, there is an urgent demand for privacy-preserving techniques capable of supporting computing and data-intensive (CDI) computing, such as training deep neural networks (DNNs) over an enormous amount of data.

TEE-based secure computing. Answering to this urgent call are confidential computing techniques, which have been studied for decades. Traditional software-only approaches such as homomorphic encryption and secure multi-party computing are considered to be less effective in protecting complicated computing (such as DNN analysis) over big data, due to their significant computation or communication overheads. Emerging as a more practical solution is the new generation of hardware supports for Trusted Execution Environments (TEEs) such as Intel Software Guard Extensions (SGX) [3], AMD Secure Encrypted Virtualization (SEV) [4] and ARM TrustZone [5]. These TEEs are characterized by their separation of a secure world, called enclave in SGX, from the insecure one, so protected data can be processed by trusted code in an enclave, even in the presence of a compromised OS and corrupted system administrators. None of them, however, can truly support CDI computing tasks, due to their exclusion of high-throughput accelerators such as graph-processing unit (GPU), tensor-processing unit (TPU), and FPGA etc. More fundamentally, today's TEEs are not designed to protect big-data analytics, since they fail to support the heterogeneous computing model that becomes the mainstream architecture for CDI computing [6]–[14]. Under the heterogeneous architecture, a CDI computing task is jointly processed by different types of computing units: e.g., a machine learning task today is typically performed by a CPU, which acts as a control unit, and a set of GPUs or TPUs, which serve as computing units. Such a joint computing model also needs to be under TEE's protection, which has not been considered in current designs.

Recent years have seen attempts to support the heterogeneous TEE. Examples include Graviton [15] and HIX [16]. However, all these approaches require changes to CPU and (or) GPU chips, which prevents the use of existing hardware, and also incurs a long and expensive development cycle to chip manufacturers and therefore may not happen in the near

[*] Corresponding authors: Rui Hou (hourui@iie.ac.cn), XiaoFeng Wang (xw7@indiana.edu)

DOI 10.1109/SP40000.2020.00054

future. Another problem is that their reliance on the host CPU TEE limits their utility. As a prominent example, SGX today has only been deployed to Xeon E3 processors, while dominant high-performance Intel CPU chips like Xeon E5 extensively used in data centers still have not included SGX instructions yet. Also, running CDI tasks and their computing supports (e.g., AI framework, GPU runtime and driver, etc.) inside a CPU TEE today significantly increases the size of Trusted Computing Base (TCB) and may not even be possible due to the resource constraints of the TEE (e.g., $<$ 100MB protected memory for SGX [3]). Finally the design based upon the enclaves of individual cores requires these enclaves to work closely with each other. This has security implications and opens a new avenue for a side-channel analysis on the communication between different enclaves.

Our new design. We believe that a TEE designed for CDI tasks should offer a strong support for heterogeneous computing, enabling collaborative computing units to be protected under a single enclave and conveniently assigned across secure/insecure worlds and different enclaves. Further this computation-oriented TEE should only include a small TCB (e.g., a single security controller with necessary code customized for supporting CDI task isolation), to reduce its complexity and also minimize the side-channel attack surface exposed by resource sharing with the untrusted OS. For ease of deployment, using existing computing units without chip-level changes is highly desirable. In this paper, we present the first design that fits all these descriptions. Our approach, called HETEE (Heterogeneous TEE), is uniquely constructed to work with today's servers, with the focus on protecting *Platform as a Service* (PaaS) against information leaks.

Unlike the existing TEE, which works on a single host, the HETEE architecture is designed to enable *dynamic allocation of computing resources for secure and non-sensitive computing tasks across multiple servers*, based upon the state-of-the-art data-center technologies such as resource pooling and PCIe switching. More specifically, running on a server rack, HETEE operates inside a tamper-resistant chassis (called HETEE box) to control a pool of *commercial, off-the-shelf* (COTS) accelerators, including GPUs, FPGAs, etc. The box is connected to other hosts on the same rack through PCIe Switch Fabric. On receiving a request from a host, the HETEE box dynamically configures the PCIe Switch to connect COTS computing units to the host, when the task is non-sensitive. For a task involving sensitive data, HETEE configures the switch to allocate computing units for a secure enclave isolated from other units, performs a remote attestation with the data owner (through one of the hosts) and then decrypts the data from the owner and runs approved code on the data inside the enclave. After the task is done, all units are sanitized and restored to their original, trusted states for processing the next task. In this way, we can leverage existing hardware to provide on-demand computing supports for both sensitive and public tasks.

Further underlying our design is the idea to *simplify TCB through cost-effective hardware design*. The HETEE architecture includes a two-level isolation mechanism based upon a Security Controller (`SC`) and a set of low-cost *microservers* that act as security `proxy nodes`. While a `proxy node` could run a complete software stack (e.g., CUDA [17], TensorFlow [18]), it is verified and protected by the `SC`, which involves only necessary functionalities like (de)encryption, remote attestation, PCIe fabric configuration, etc., before touching sensitive data. Isolation between the HETEE and the outside world and among different enclaves is achieved *physically* with the PCIe switch and the proxies, each controlling a separate enclave. Restoration to secure states for the `proxy node` happens through secure reboot of the server. This design avoids software-based enclave control, isolation, etc., thereby reducing the TCB size.

We performed a security analysis of HETEE and further implemented it on a PCIe ExpressFabric backplane, CPU+FPGA (for the `SC`), Intel Xeon E3 based microservers (for the `proxy nodes`), and 4 Nvidia TITAN X GPUs, with the TCB including only necessary security and management code, excluding the heavy software stack for controlling accelerators or executing the AI runtime. Running the implementation on DNN training and inference tasks of the real-world scale (152-layer ResNet network, with 60 MB parameters and 200 MB model size on the $\sim$138GiB ImageNet data set), we observed an average 1.96% throughput overhead and 34.51% latency overhead for inference, 0.60% throughput overhead and 14.24% latency overhead for training.

Contributions. The contributions of this paper are summarized as follows:

• *New TEE design for scalable confidential computing*. We present the first design for data-center level TEE, supporting super large-scale confidential computing. Our design leverages the state-of-the-art computing unit pool technologies to dynamically allocate computing resources for both secure and non-sensitive computing tasks across all servers on a rack. It further reduces security risks using a centralized yet cost-effective HETEE box (with the expense of confidential computing hardware below 5% of the cost of computing units), and hardware-based TCB simplification.

• *Implementation and evaluation*. We implemented our design and evaluated it on large-scale DNN training and inference tasks. Our study shows that our approach largely preserves the performance of heterogeneous computing in the trusted execution environment, which has never been achieved before.

Roadmap. The rest of the paper is organized as follows: Section II presents the background and threat model; Section III provides the HETEE design; Section IV and V describe the HETEE prototype system and typical confidential AI computing services; Section VI reports the performance evaluations; Section VII elaborates our security analysis and Section VIII is the discussion; Section IX surveys the related works and Section X concludes the paper.

II. Background

A. Heterogeneous Data-Center Computing Architecture

Heterogeneous computing support. Heterogeneous computing architectures are commonly used in data centers, since a large-scale computing task such as DNN training and inference often needs to be processed jointly by different computing units (GPUs, other accelerators, etc.). Since configuring hardware systems, particular expensive computing units like GPUs, based on peak workload usually leads to over-provision, which increases cost, on-demand resource allocation and cost-effective architecture are becoming mainstream [74]–[79]. Serving this purpose is a *resource-pooling* technique (aka., resource disaggregation) powered by the PCIe switch network, which is known to be a mature solution for building high-density data center servers [28] [29]. For example, Facebook's AI hardware acceleration systems (Zion and Kings Canyon [6]), Nvidia's DGX-2 [19] [20] and HGX-1/2 series [21]–[24] all utilize PCIe ExpressFabric to construct their heterogeneous architectures. The PCIe ExpressFabric chips enable a group of CPUs to flexibly share multiple GPUs and other high-performance IO devices, thereby reducing cost and increasing resource utility.

PCIe ExpressFabric. Besides the capability of traditional PCIe switch chip, PCIe ExpressFabric chip is also featured with two unique properties important to our design:

- *Software-defined fabric*. The switch is built on a hybrid hardware/software platform that offers high configurability and flexibility with regards to the number of hosts, end-points, and PCIe slots. Its critical pathways have direct hardware support, enabling the fabric to offer non-blocking, line speed performance with features such as I/O sharing. The chip has a dedicated port for management, through which an external management CPU (mCPU) can initialize the switch, configure its routing tables, handle errors, Hot-Plug events, and others. In this way, all the hosts connected by the switch only see what the mCPU allows them to see.

- *Flexible topology*. The switch eliminates the topology restrictions of PCIe. Usually, PCI Express networks must be arranged in a hierarchical topology, with a single path from one point to another. ExpressFabric allows other topologies such as mesh.

B. Trusted Execution Environment

A trusted execution environment (TEE) guarantees that the code and data loaded into an isolated area (called an enclave) are protected to ensure their confidentiality, integrity and authenticity. TEE is designed to thwart not only the OS-level adversary but also the malicious party who has physical access to the platform. To this end, it offers hardware-enforced security features including isolated execution, integrity and confidentiality protection of the enclave, along with the ability to authenticate the code running inside a trusted platform through attestation:

- *Isolation*. Data within the enclave cannot be read or modified by untrusted parties.

- *Integrity*. Runtime states should not be tampered with.

- *Confidentiality*. Code, data and runtime states should not be observable to unauthorized applications.

- *Authentication*. The code under execution has been correctly instantiated on a trusted platform.

Existing TEEs, including Trustzone and SGX, and the solutions that extend CPU TEEs to protect heterogeneous units [15] [16] are focused on protecting the operations of individual computing units, not their high-performance interactions. Such designs expose a large surface to side channel attacks during heterogeneous computing and also increase overhead when computing results move from one enclave to the other. As a result, they are less suitable for supporting super large-scale confidential computing.

C. Threat Model

We consider a strong adversary who controls the entire software stack on host systems and has physical access to the HETEE platform, as elaborated below:

(Privileged) software adversary. HETEE defends against the adversary with full control of the software stack on the host systems, including unprivileged software running on the host and the host OS. Such an adversary can also mount a side channel attack e.g. by analyzing network traffic. Covert channels are currently out of the scope of the paper.

Hardware adversary. An adversary with physical access to the server can mount snooping attacks on the host memory bus. We assume that the adversary cannot physically tamper with the HETEE box, as the box used in a data center can be a secure self-destructing chassis that is armed with a micro-controller (MCU) system and a set of sensors (e.g., pressure, vibration and temperature etc) for access control management and intrusion detection/response [30]–[34], [104], [105]. As such, the hardware adversary cannot mount a snooping attacks on the PCIe fabric within the HETEE box. We exclude electromagnetic and power analysis and leave them to the future work. Like TrustZone, our approach is not completely immune to a cold boot attack, but does provide a certain level of protection: the cold boot attack [98]–[100] cannot succeed when the time taken to open the sealed box illegally is longer than that for retaining memory content after the power is removed. Also if necessary, the HETEE chassis could include more expensive self-destructive protection [101], [102].

Others. We use standard cryptographic techniques, and attacks against cryptographic algorithms are out of our scope. The ciphertext communicated between HETEE and remote users needs to be forwarded by non-secure hosts, so denial of service attacks are also not considered in this paper. We also trust the FPGA synthesis tool and the mCPU firmware. We assume that the firmware of GPU does not include malicious code and its integrity is protected, and our design has to trust the hardware vendor for the correctness of the firmware updates (see Section VII).

III. HETEE Architecture

A. Design Overview

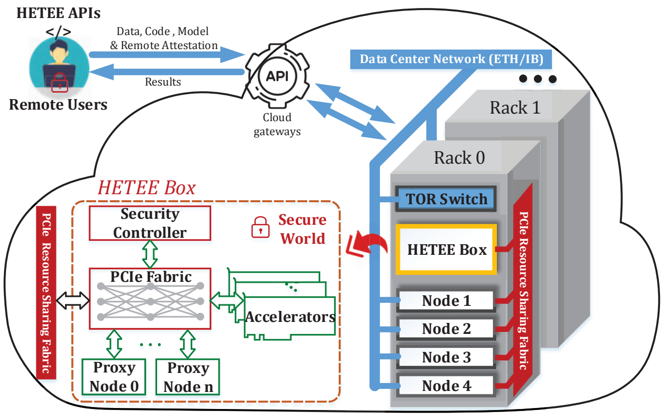


Fig. 1. HETEE Overview.

Idea and architecture. HETEE is designed to provide an efficient, practical and flexible trusted execution environment for rack-scale heterogeneous computing in clouds and data centers, with the focus on protecting *Platform as a Service* (PaaS) (Machine Learning Platform as a service (MLPaaS) [6], [7], [109] in particular) against information leaks (to other platform users and the platform provider). Through HETEE, the cloud service provider can create an enclave for a user on a proxy node, running OS and other public platform software (e.g, Tensorflow), while the enclave user uploads her code and data into the enclave to run her task on the platform (e.g., training a ML model). The result will only go back to the user. This service model has been used by today's TEE service providers such as Microsoft Confidential Computing [109]. Compared with prior work [15] [16], our approach is unique in that it leverages existing data center technologies and resources to achieve scalable protection, requires no chip-level change, and strives to minimize the side-channel attack surface.

Fig. 1 illustrates the architecture of a typical data center running HETEE. Our approach uses PCIe ExpressFabric as a high-speed, low-latency resource sharing network inside the rack, connecting local computing server nodes to a pool of heterogeneous computing units (GPUs, FPGA-based accelerators, etc.). In particular, the HETEE box provides the rack-scale resource sharing fabric built on PCIe Expressfabric chips. While other commercial computing nodes may only integrate traditional PCIe transparent switch chip rather than the PCIe Expressfabric chips for I/O extensions, they can connect to the resource sharing fabric via PCIe extension adaptor and PCIe cable. Inside each rack, the HETEE box manages heterogeneous units, dynamically allocating them to computing tasks and isolating them from each other through several modules, including *Security Controller* (`SC`), `proxy nodes`, and accelerator resources with PCIe interfaces.

User support. HETEE provides a set of APIs and a library for remote users to utilize the TEE service. Through these toolkits, the user firstly establishes a trust relation with an HETEE enclave (through its `SC`) and negotiates with it a shared secret through a remote attestation. The subsequent messages between the user and the box are then encrypted and integrity-protected.

For this purpose, the APIs we provide include the typical functions `send_message` and `receive_message` that deliver the following three messages based on a classic request/acknowledgement protocol:

- *Configuration messages*: The remote user sends the configuration message to the `SC` to create a new HETEE enclave, as well as the specified type and number of accelerators. The `SC` will assign a unique ID to each enclave.
- *Code messages*: These messages are used to transfer programs to be executed to the HETEE enclave, which could be AI models in the ONNX format [35], or CUDA code etc.
- *Data messages*: The messages used to deliver sensitive data.

Design challenges. Behind the support for the remote user are a set of techniques developed to securely and dynamically share computing resources for both secure and non-secure computing tasks. These techniques are meant to address the following technical challenges: (a) how to share computing resources while providing strong isolation for HETEE enclaves (Sec. III-B); (b) how to minimize the TCB (Sec. III-C).

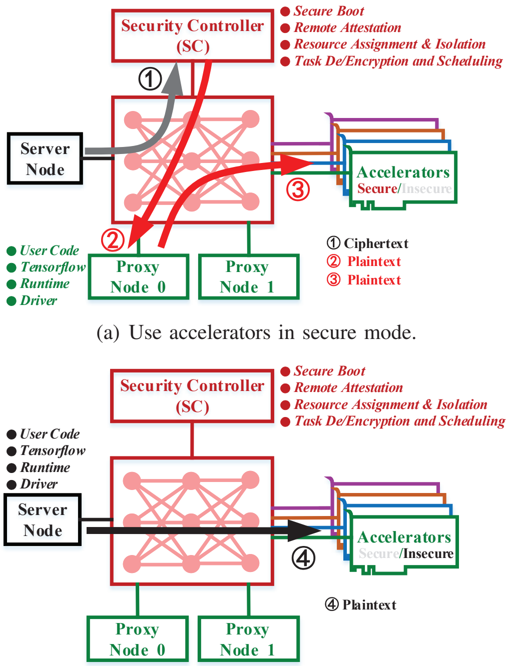
For (a), HETEE utilizes the PCIe ExpressFabric to dynamically and physically isolate an enclave from other enclaves and from the untrusted OS. For (b), we adopt a unique two-level isolation strategy in which the `SC` is the only trusted node and runs a small set of firmware with integrated security and management code, while the `proxy node` operates the heavy software stack for controlling accelerators and executing the AI runtime. This approach, together with our use of hardware to replace software control, helps simplify the TCB.

In the rest of the section, we present the detailed designs of these techniques, together with the mechanism to establish trust between the HETEE and the user.

B. Elastic Resources Allocation and Isolation

Elastic resources allocation. The PCIe ExpressFabric Switch chip has a dedicated management port for configuration, which is used by the `SC` for computing unit allocation and computing isolation. Through the chip's driver, the `SC` can implement a PCIe network configuration using its APIs and/or Command Line Instructions (CLIs). This allows definition of different connection topologies on-demand, so as to dynamically assign accelerators to the hosts on the same rack and separate different computing tasks from each other. More specifically, such elasticity in resource allocation and isolation offers the following supports:

- *Elastic allocation of pooled secure accelerators*. Pooled accelerators in the secure state can be dynamically allocated as security resources: that is, the `SC` can assign multiple accelerators to a dedicated `proxy node` that runs an enclave by configuring the PCIe fabric chip. As shown in Fig. 2 (a), to handle a secure computing task, a server node forwards encrypted requests and data from the user to the `SC` first, which then decrypts the messages and delivers the content to the `proxy node` that controls secure accelerators. Through the



(a) Use accelerators in secure mode.

(b) Use accelerators in insecure mode.

Fig. 2. Elastic allocation of resources.

dynamic network configuration, we can allocate accelerators in the resource pool according to computing requirements, achieving better resource utilization and efficiency. The remote users can request accelerators for their tasks. An AI framework can also automatically assess its workload and ask for an appropriate amount of the accelerator resource.

- *Secure mode switch across secure/insecure worlds for accelerators*. HETEE is also designed to support the dynamic switch of the accelerator between secure and insecure worlds, to share the expensive computing unit across different servers when they work on non-sensitive computing tasks. Under the HETEE architecture, computing units for the servers on the same rack are all managed by the HETEE box. They are dynamically allocated to the nodes outside the box through reconfiguring the PCIe switch network, so the nodes can directly manage and use the accelerator shown in Fig. 2 (b). When this happens, however, the accelerator moves from a secure state to an insecure one. So when the unit comes back to the pool for a sensitive task, it needs to be restored to a secure state.

To support the mode switch of accelerators, we have designed a priority-based preemptive accelerator scheduling mechanism. A secure switching service needs to run on the SC. In the meantime, local nodes (i.e., the servers on the same rack) can ask the controller to release some of the accelerator resources from the secure world through a configuration message or an out-of-band request. The switching service can also send high-priority requests to local nodes, halting the tasks performed by the accelerator in the insecure world and bringing them back to the secure world.

Efficient secure cleanup. When an enclave is destroyed, both the proxy node and related computing units need to be cleaned up to remove data and restored to the "secure" state before the establishment of a new enclave. For this purpose, the SC initiates a cold, secure reboot on the proxy node, which clears the context including all the data inside memory and architectural registers. Meanwhile, all accelerators assigned to the proxy node are also powered off and reset to get back to their original, secure states, assuming that their firmware has not been compromised. Our experiments show that such cold reboots can effectively remove the memory and accessible registers' content on the state-of-the-art GPUs like NVIDIA TITAN X and NVIDIA Tesla M40.

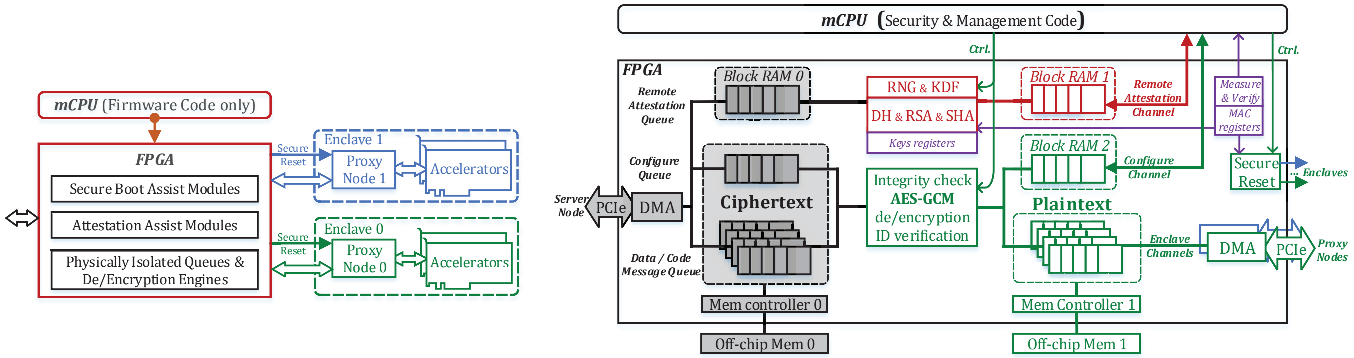
The secure reboot process ensures that the proxy node can only load the OS and accelerator software from signed images on the SC. This has been done by modifying the microserver's PCB board. Specifically, we removed the Boot ROM chip on the proxy node board and connected the SPI interface circuit line for boot-loading to the dedicated IO pin of the FPGA chip on the SC board. As a result, the function of the Boot ROM chip is replaced by the module on the FPGA chip, which verifies and loads the OS and other code to the proxy node. Also, taken over by the SC board is the Intelligent Platform Management Interface (IPMI) physical interface on the microserver that is used to remotely manage and control firmware and system updates.

A problem here is that secure rebooting takes a relatively long time (tens of seconds) and might affect the HETEE's response time and throughput. In practice, however, the issue is less of a concern, due to the HETEE's use of a low-cost high-density microserver cluster, with 2 proxy nodes being integrated on each adapter card (two cards in our prototype, see Figure 5(d)). Once rebooted, a node with clean context is registered to the idle queue. When a new task arrives, the SC takes the first node in the queue to create an enclave for the task, together with required accelerators. This approach helps reduce the average waiting time when the task arrival rate is relatively low. Since a HETEE box typically carries no more than 32 GPUs to serve no more than 20 servers on the same rack, 4-8 microservers, each managing a single enclave, should be adequate, as implied by the observation in the prior research [37]–[41]: the workloads of a data center are generally characterized by a diurnal pattern (leaving the accelerators and servers under-utilized for most of the time except peak hours). The system's scalability can be improved using the techniques like virtualizing TrustZone [42]. However, this software based isolation increases the complexity of security control, which might lead to a thicker TCB. How to further improve resources utilization is left to our future research.

C. Hierarchical Isolation and TCB Simplification

Running a complicated computing task inside HETEE often needs the support from a complicated software stack. For example, to perform DNN training or inference, we have to load AI runtime (TensorFlow [18], Caffe [43] [44] or PyTorch [45] [46]), the GPU runtime and driver (CUDA) to an enclave.

To address the security risk, we come up with a two-level isolation design that simplifies the TCB software stack. Only security and management modules are deployed to the SC,



(a) Structure of the SC and HETEE Enclaves.

(b) SC modules.

Fig. 3. Security Controller. (RNG: Random Number Generator; KDF: Key Derivation Function; MAC: Message Authentication Code; DH: Diffie-Hellman key exchange engine; RSA: public key signature/encryption engine; SHA: Secure Hash Algorithm engine; AES-GCM: symmetric de/encryption engine;)

which form a thin TCB for the HETEE (our implementation includes isolation and security management modules on open-source coreboot [87] firmware). Other software components, including the GPU driver, runtime, and the AI framework like TensorFlow, run on the proxy node. As mentioned earlier, the proxy node is *outside* the TCB, physically and logically separated from the SC but controlled by the SC. When computing a task, each proxy node manages an enclave and all its computing units but has been isolated from the outside world by the SC. The integrity of its initial software stack is verified by the SC and proven to an enclave user (the data owner) through attestation and the computing results are only sent back to the user. The physical isolation of the enclave reduces resource sharing with other enclaves and the untrusted host, minimizing the side-channel attack surface. Once the computation is done, the node is sanitized and restored to the trust state through secure rebooting (Sec. III-B Efficient secure cleanup), together with all computing units assigned to the enclave. Below we elaborate the design and implementation of these components in the HETEE.

Security controller. As a separate system, the security controller runs on a board connected to the standard PCIe fabric, acting as a gatekeeper for the secure world inside HETEE. The SC can be implemented in a variety of ways, using CPU, CPU+FPGA, or a custom ASIC chip. Since CPU is not as efficient as FPGA when performing encryption and authentication computation, and the cost of developing the ASIC chip is high, we utilized CPU+FPGA to built a prototype (Fig. 3). The main functionalities of the SC are as follows:

- *Secure boot*: Secure boot is performed by the measure module on FPGA and the firmware code of mCPU, which is responsible for the security of the SC itself, as well as restoration of the proxy node's trusted state.

- *Resource assignment*: This module is a program integrated into the mCPU firmware. It is designed to dynamically configure the PCIe fabric according to security or non-security requirements, for the purpose of assigning accelerators to computing nodes, isolating resources for enclave management and efficient security state switching for pooled accelerators.

- *Remote Attestation*: The attestation module is implemented inside the FPGA chip to support cryptographic operations (key establishment, authentication, etc.) for establishing a trust relation between the remote user and the HETEE box. The details are presented in Sec. III-D.

- *Message (De)Encryption/Parsing/Scheduling and hardware-assisted isolation and control*: This module is implemented inside the FPGA chip for recording, (de)encryption and scheduling of messages and controlling the access to them from different enclaves. It is designed to isolate enclaves, particularly the proxy nodes operating them, from the outside world (the host nodes, remote users, etc.). This is achieved through hardware-assisted separation, such as the use of private DDR4 controllers with hardware-wired isolated access paths of logic implementation inside the FPGA.

The hardware-based message isolation and control is shown in Fig. 3 (b). Again, the mechanism, together with the resource assignment module, is designed to physically separate the enclaves inside the HETEE box and the outside world, so the SC can have full mediation on the interactions between the secure and insecure worlds. In the meantime, the design is meant to minimize the attack surface on the SC, to protect its (already thin) TCB from potential exploits. Specifically, the isolation and control mechanism is made very simple and fully implemented in hardware, on the FPGA board that is separated from the mCPU board running firmware-based security management of the SC. This minimizes the threat from the input data uploaded by untrusted sources. Further the untrusted server nodes on the rack can only access the hardware queues that store encrypted messages (i.e., the ciphertext data/code message queue and the ciphertext configuration queue, see the grey box in Fig. 3 (b)). Those queues are logically mapped to each node's own PCIe space by the PCIe ExpressFabric chips. The FPGA logic does not implement the path for an external node to access plain-text messages in another hardware queue (i.e., the plaintext configuration queue), essentially eliminating the possibility for out-of-bound access. Similarly, the proxy node inside the HETEE box can only access the plain-text hardware message queues (i.e., the plaintext data/code message queue). The ciphertext queues and the plain-text queues are built in the FPGA programmable logic, each entry of which contains pointers to specific messages that are stored in two physically isolated DDR memory blocks.

The FPGA board assigns a pair of queues to each enclave, one for encrypted data (in the grey box) and the other for the corresponding plaintext (in the green box). These queues are physically separated from other queues on the board. Only the isolation and control mechanism, which is also fully implemented in hardware, is allowed to access queues on both sides. It runs AES to encrypt/decrypt the data from one queue before moving the result to the other queue. The module also performs authentication (random ID in the encrypted message) and integrity check (AES-GCM) to ensure that an enclave only receives correct messages from the authorized party (the one uploading the computing task in our research).

Proxy node. The `proxy node` is also a stand-alone system in the HETEE box and connected to the PCIe fabric. Each enclave runs on a separate `proxy node`. This reduces the surface for possible inter-task information leaks. During task switching (or enclave destroy), the `proxy node` is forced to reboot by the `SC` to load trusted images and reset its state.

Inside the HETEE enclave, the `proxy node` is responsible for managing the accelerators dynamically allocated to the enclave by the `SC`. It runs a simplified Linux and the user-level software stack for accelerator and a typical AI runtime. Note that the integrity of this stack is ensured by the `SC` once the `proxy node` is rebooted and also the stack only serves the current computing task. In addition, during the computation, the `proxy node` cannot interact with the outside without the mediation from the `SC`. This reduces the surface the software stack exposes to the adversary, even when it contains vulnerabilities. Also under the control is the side-channel attack surface: unlike the prior approaches [15], which require a CPU TEE (like TrustZone or Intel SGX) to work with a GPU (possibly through its own TEE [16]), computing units and the `proxy node` all run inside the same enclave, which allows them to interact with each other in a highly efficient and also secure fashion, with the communication between the server and the units all hidden from a side-channel attacker.

The `proxy node` accepts the code and data messages dispatched from the `SC`, runs the AI model written by the remote user on its local AI software stack, and analyzes the data contained in the data message by invoking the assigned accelerators. The results are then packaged into a result message sent to the `SC`, and eventually delivered to the remote user through an encrypted channel.

As the major workloads on the `proxy node` usually include the AI runtime as well as different AI models, and the intensive computation is offloaded to the accelerators, the `proxy node` needs CPUs with strong I/O capabilities, even when their computing capabilities are relatively weak. Therefore, in our research, we chose microserver as the `proxy node`, which integrates low-end Xeon E3 processors. This design can save both cost and power consumption, which in turn increases integration density, allowing more microservers to be put on a single board.

D. Trust Establishment

Secure Boot. The booting of HETEE starts from the chip boot circuit on the FPGA, which loads the encrypted bitstream file from the external boot flash memory. Then the measurement logic on the FPGA verifies the integrity of the mCPU firmware (with the security management code). After the `SC` subsystem boots up, it evaluates the firmware of `proxy nodes`. Once its integrity is confirmed, the firmware proceeds with a typical secure boot process to check the proxy's kernel image, which further verifies the binaries of critical applications.

Remote attestation and key negotiation. Remote attestation of HETEE follows the standard protocol used by SGX [92]–[94] and Sanctum [65], [66], which is a combination of the SIGMA [90] authenticated key exchange protocol and TCG's attestation protocol [103]. It supports mutual authentication and establishment of a secure channel between the remote user and a HETEE enclave. SIGMA is a popular key exchange protocol that has undergone rigorous security analysis [91]. Like SGX and Sanctum, we integrate the `SC` measurement and the enclave measurement (with the version number of the `SC` bitstream, firmware, and `proxy node` software) into the protocol. Since an enclave is created through a secure reboot, its software state is measured with the hash values of the kernel and applications verified during the boot.

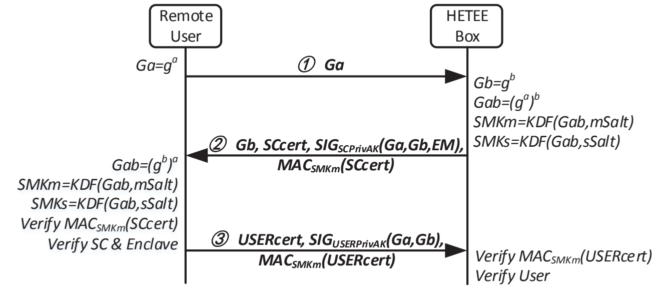

Remote User
HETEE Box
$Ga=g^a$
① Ga
$Gb=g^b$
$Gab=(g^a)^b$
SMKm=KDF(Gab,mSalt)
SMKs=KDF(Gab,sSalt)
② Gb, SCcert, $SIG_{SCPrivAK}$(Ga,Gb,EM), MAC_{SMKm}(SCcert)
$Gab=(g^b)^a$
SMKm=KDF(Gab,mSalt)
SMKs=KDF(Gab,sSalt)
Verify MAC_{SMKm}(SCcert)
Verify SC & Enclave
③ USERcert, $SIG_{USERPrivAK}$(Ga,Gb), MAC_{SMKm}(USERcert)
Verify MAC_{SMKm}(USERcert)
Verify User

Fig. 4. Remote Attestation and Symmetric Key Negotiation Protocol. (*Each HETEE platform includes two sets of public key pairs, an Endorsement Key (EK) and an Attestation Key (AK) (see Sec. III-E). The SC_{cert} contains the SC measurement and is signed using the EK private key. The enclave measurement is signed with the AK private key. The trust of AK and EK is derived from the certificate chain generated from the HETEE vendor's root key and can be verified by the user. EM: enclave measurement; KDF: Key Derivation Function; MAC: Message Authentication Code; mSalt and sSalt are cryptographic random numbers; Ga, Gb and Gab are used for the Diffie-Hellman key exchange. USERcert is the user's public key certificate.*)

The HETEE attestation protocol is shown in Fig. 4. Message ① and message ② implement Diffie-Hellman key exchange. Message ② is used by the HETEE platform to send its certificates and report measurements to the remote user for identification and authentication. It also carries a message authentication code $\mathrm{MAC}_{\mathrm{SMKm}}$(SCcert) for the remote user to check its integrity. Similarly, message ③ is for the remote user to send her certificate to the HETEE platform, which also includes $\mathrm{MAC}_{\mathrm{SMKm}}$(USERcert) for integrity check.

Similar to other TEE techniques (such as Privacy CA of TCG [103], [115], Sanctum [65], [66], Graviton [15], SecTEE [117] and Keystone [64], [114]), the proposed attestation protocol requires trusted CA for secure certificate distribution. Intel SGX adopts Enhanced Privacy ID (EPID) [93], an enhancement of the Direct Anonymous Attestation (DAA) algorithm [116], which does not need trusted CA but still relies on a trusted party (i.e., Intel) as the Issuer for key distribution.

Our current design assumes that certificate distribution of the attestation protocol is secure. Prior research shows that both CA-based and EPID-based protocols are vulnerable to the Cuckoo attack [118] in which the adversary relays the attestation messages to a TEE under his physical control. The attestation of the HETEE box is implemented on SC, which is included in our TCB and protected from physical attacks for controlling the box (see Sec.VII). Also, solutions have been proposed for mitigating the threat of the Cuckoo attack [119], [120], which could be incorporated into our attestation protocol.

E. Key Management in SC

Endorsement Key (EK) and Attestation Key (AK) for each HETEE platform. Each HETEE platform includes two sets of public key pairs, an Endorsement Key (EK) and an Attestation Key (AK). These keys are the basis to authenticate the platform, and will never leave the FPGA chip at any time. The private key of EK and the EK certificate are encoded by the vendor into the encrypted FPGA bitstream. The EK is only used by a dedicated signing logic to sign the AK and the SC measurement for creating an AK certificate. The AK is derived from the random number generator module on the FPGA each time it is powered up, and the private key of the AK and the AK certificate can only be accessed by a dedicated attestation logic module which is specifically used to sign Diffie-Hellman parameters and enclave measurements, for remote attestation and authentication. The EK is only used when generating the AK certificate, and all credentials for remote attestation are generated using the AK. This certificate chain design limits the usage of the EK, thereby reducing the risk of its exposure, since the key is the root of trust for the platform.

Two sets of symmetric keys for each User. SMKm and SMKs are generated during remote attestation. The SMKm is used for integrity check with MAC (message authentication code) during remote attestation. The SMKs is used to build the secure channel between the user and the HETEE box.

Symmetric key for hard disk. The SMKdisk is used for data protection inside the HETEE box. That is to say, all the data on the internal disks of the HETEE box are encrypted. The (de)encryption engine locates in the disk controller. Note that commercial SSD controllers usually have such an engine for line-speed data protection. The SC FPGA generates the SMKdisk and sends it to the disk controller. The key is not stored on the disk by the disk controller, and instead is thrown away when the system is power off for data protection.

F. Sealing, Management and Maintenance of HETEE Box

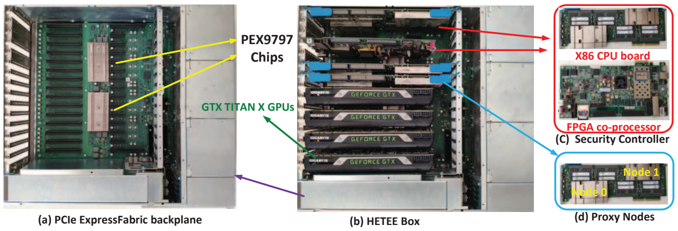
Sealing. The chassis of the HETEE box is designed to be tamper-resistant, based upon the techniques used by proven products [97], [104], [105], [108], which can meet the NIST FIPS 140-2 security Level 3 or 4 [106], [107]. A typical implementation includes a microcontroller (MCU) system and a set of sensors (e.g., pressure, vibration and temperature etc) for access control management and intrusion detection/response. The HETEE box provides a USB interface for authentication based on the USB key. When the chassis is opened without authorization, the pressure sensor notifies the MCU. Then the MCU actively empties the bitstream located in the FPGA-connected flash chip and power down the entire system. When powered up again, the FPGA's security boot mechanism detects such exception through verification, even when the attacker replaces the bitstream. In this case, the FPGA module will stop booting. The attacker cannot get sensitive content from the volatile memory, and nor can he touch the content of the encrypted data on the disk since the disk key is destroyed.

The HETEE box can be sealed by the system vendor, just like HSM devices, when the data center is not trusted with the data it processes. Alternatively, this can be done by the authorized party in the data center, to prevent the access from unauthorized members and mitigate insider risks.

Maintenance. If the HETEE box has a hardware failure or needs to upgrade its software (e.g., firmware on SC, AI runtime and Linux on proxy node, as well as GPU firmware), an authorized administrator can open the chassis properly for maintenance using the USB key. Here the administrator can be the vendor of the HETEE box, or the third party such as the authorized member in a data center who is allowed to access the protected data. Anyone else even the cloud provider is not permitted to open the chassis.

Cooling. The HETEE box uses a standard server blade chassis enhanced with protection against unauthorized access (as described before), which supports high-performance server level cooling mechanisms, such as air channel design, front/rear thermal vent, fans with speed control, air or water cooling, which are common to the server system. This enables the HETEE box to be easily deployed in data center.

IV. HETEE PROTOTYPE SYSTEM



(a) PCIe ExpressFabric backplane (b) HETEE Box (C) Security Controller (d) Proxy Nodes

Fig. 5. HETEE prototype system.

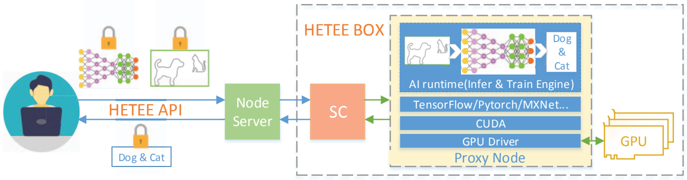


Fig. 6. Confidential AI service provided by HETEE system.

The HETEE prototype is shown in Fig. 5. The HETEE box (Fig. 5(b)) uses two Broadcom PEX9797 chips to build the PCIe ExpressFabric backplane shown in Fig. 5(a), which has 15 PCIe slots. As an industry-leading ExpressFabric platform, the PEX9700 series switch chips provide high performance, low latency, and scalable and cost-effective connections based on the PCIe Gen3 protocol. They also provide the ability to share I/Os, and to enable multiple hosts on a single PCIe-based network using standard PCIe enumerations. PEX9797 has the top I/O capabilities in its series [25].

The current HETEE box includes 1 `SC` node, 4 `proxy nodes`, and 4 Nvidia GTX TITAN X GPUs. Both the `SC` and the `proxy nodes` are implemented as custom boards shown in Fig. 5(c)(d), each of which is an independent system. These boards and GPU cards are plugged into the PCIe slots of the PCIe Fabric backplane. These nodes can communicate through either Tunneling Window Connection (TWC) or ethernet-like DMA as standard hosts and/or end-points.

TABLE I
HETEE PROTOTYPE SYSTEM.

Component	Hardware	Software
Security Controller	Intel Xeon-E3 1220V6 DDR4 16GB 2400MHz Xilinx Zynq FPGA	Tailored coreboot 4.10 with security management code, binary size is $<$ 300KB
Proxy Node	Intel Xeon-E3 1220V6 DDR4 16GB 2400MHz	TensorFlow 1.11.0 CUDA 9.0 Nvidia Driver 396.54 CentOS 7.2
GPU	Nvidia GTX TITAN	
PCIe Fabric	PEX9797	

Table I describes the major components of the `SC` and the `proxy nodes`. The `SC` includes an Intel Xeon-E3 CPU, which runs the security management functions (configure PCIe expressfabric chip) on the customitzed open-source firmware (coreboot [87]) and an Xilinx Zynq FPGA card, which is in charge of the remote attestation and (de)encryption. The `proxy node` integrates an Intel Xeon-E3 CPU and runs a complete GPU software stack and Tensorflow 1.11.0.

To accommodate the standard Nvidia GPU acceleration card (3U height), the HETEE box is currently a 4U and half-length chassis. The height and length of the chassis could be expanded to integrate more accelerators with more PEX9797 switching chips. This prototype system does not include tamper resistance functionality as we use it mainly for performance evaluation in this study.

V. Confidential AI Computing Service

The HETEE system enables confidentiality for applications requiring privacy. Without losing generality, this paper shows a case of safeguarding confidential AI models for AI services running on top of a pool of GPU resources as shown in Fig. 6. We used the popular ONNX open format [35] to describe AI models. With ONNX, AI developers can easily move models from one state-of-the-art tool to another and choose the combination that is best for a target application.

AI inference. For an inference task, the remote user first authenticates and exchanges keys with the HETEE box, establishing a trusted relationship and obtaining a shared symmetric key for secure communication. The remote user encrypts the AI model with the shared symmetric key and sends it to the HETEE box. Upon receiving the encrypted AI model, the `SC` decrypts it and sends it to the AI runtime (TensorFlow in our prototype system) on the `proxy node`. The AI runtime parses the model and starts the corresponding inference engine. The remote user then can begin to send encrypted data to the HETEE box in batches. The inference engine gets the data from the `SC` which performs data decryption, and invokes the corresponding GPU kernel to process the received data.

AI training. Similarly for a training task, after the user authenticates and exchanges keys with the HETEE box it sends in the initial neural network structure and also provides additional training-related configurations such as hyper-parameters (such as learning rate, epoch, batch size, etc.) and selected optimization algorithms (such as SGD, ADAM, etc.). After receiving the message, the `SC` starts the AI runtime to build the training engine. The remote user then begins to move a stream of encrypted data to the HETEE box. The training engine on the `proxy node` performs the forward pass, the backward pass, and so on for each batch of received data, and returns the current loss and part of the intermediate network state to the remote user in the encrypted form. The return values are evaluated by the remote user to determine the convergence of the network. The training process continues until the network converges or diverges. The remote user then sends out a termination command to inform the training engine that the training task is finished.

VI. Evaluation

A. Methodology

AI workloads. We trained 5 classical neural networks on ImageNet 2012 [53], which contains millions of images in 1000 categories. We used 138 GiB training images of ImageNet 2012 as our training dataset, while 6 GiB of images were used as validation dataset. Besides, the test data set for inference included 13 GiB of images. The number of layers of the tested networks ranged from 16 to 152, with the number of parameters between 5 million and 138 million. Our choice covered typical neural networks from small scale ones to large ones, with 5 different batch sizes selected as network inputs for each network. Table II summarizes the details of each model. We ran the inference and training workloads on our HETEE prototype described in Sec. IV.

Since the HETEE system only affects the communication path and software stack partition, it does not change the existing algorithms. Therefore, the HETEE system does not have any impact on the classification accuracy, thus our experiments mainly focus on evaluation of the throughput and latency, as well as the overhead of hardware modules such as data encryption and decryption.

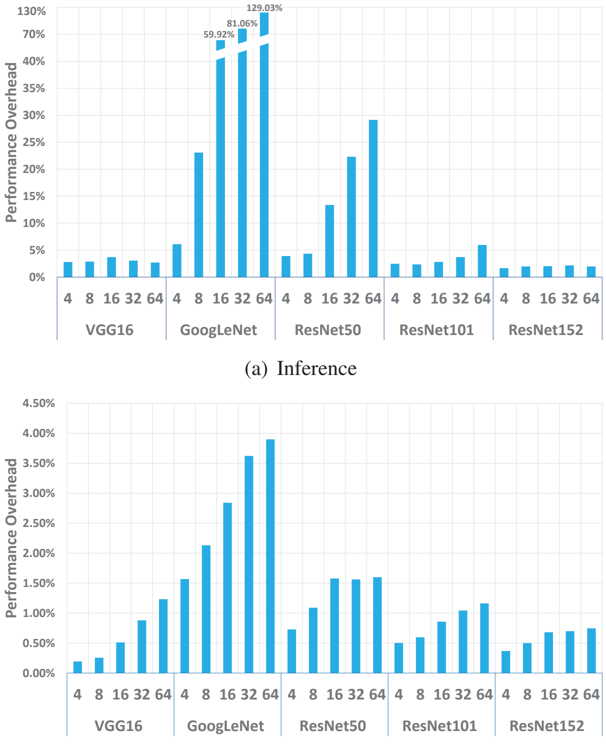
B. Performance analysis

Compared to a standard GPU server (X86 CPU + GPU), the HETEE system provides confidential computing services while inducing longer message transfer paths, additional data

TABLE II
AI WORKLOADS.

Model	Model size	Parameters	Layers	Image size	Type
VGG16 [54]	500 MiB	138 M	16	224x224x3	1K
GoogLeNet [55]	28 MiB	5 M	22	224x224x3	1K
ResNet50 [56]	100 MiB	25 M	50	224x224x3	1K
ResNet101 [56]	150 MiB	44 M	101	224x224x3	1K
ResNet152 [56]	200 MiB	60 M	152	224x224x3	1K

forwarding and pre-processing. Here we report a study on the throughput and the latency overhead of our implementation by running a set of DNN workloads (Table II), as well as its scalability under the elastic resource allocation. For this purpose, we utilized as a baseline the same workloads' performance on a standard GPU server, with the same GPUs.



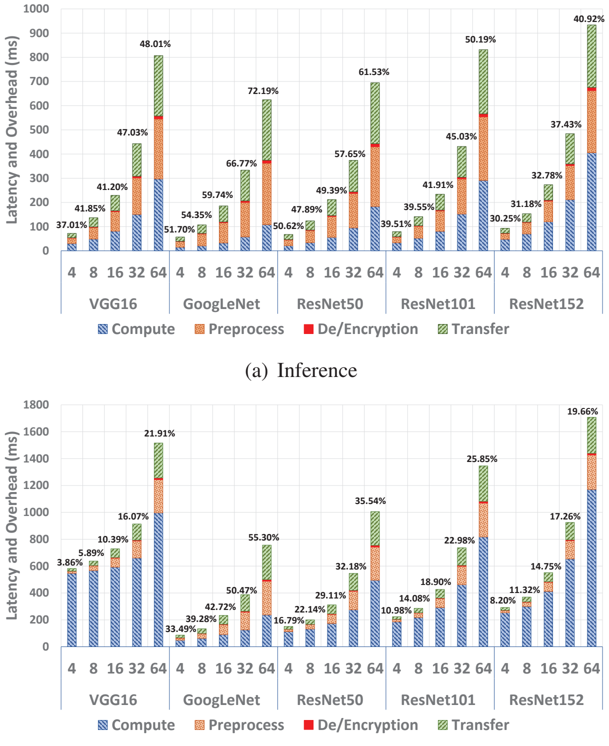
(a) Inference

(b) Training

Fig. 7. HETEE throughput overhead on single GPU with different batch sizes.

Throughput and latency evaluation on a single GPU. The throughput overhead of the HETEE system normalized to the baseline is shown in Fig. 7. It can be seen that the throughput overhead is 6.95% for the inference task and 0.91% for the training task on average when the batch size is 8. In this case, the GPU utilization is around 80%. For most of the workloads in our evaluation, the throughput overhead is under 5%. Such results demonstrate that the software and hardware co-design of HETEE is balanced. One exception is the GoogLeNet when it was used for inference, since the size of GoogLeNet is much smaller than others (its network model is only 28 MiB, and only has 22 layers). The smaller computing time amplifies the cost of data transmission, thereby affecting the throughput. Such an impact of model size on throughput can also be observed from ResNet, when its layers grows from 50 to 152.

Fig. 8 describes the latency overhead of the HETEE system. The Y axis shows the latency and its breakdown, and the number on top of each bar is the latency overhead normalized to its baseline. When the batch size is 8, the average inference latency is 42.96% compared to the baseline, and the average training latency is 18.54%. In this case, the GPU utilization ranges from 85% to 92%. A typical confidential DNN task running in HETEE takes 4 major steps, including pre-processing (image decoding), GPU execution, (de)encryption, and data transfer (from local node to SC, and from SC to proxy node etc). With the same delay caused by pre-processing and GPU execution as that of the baseline setting, HETEE introduces a new overhead for (de)encryption and additional data transfer. As we can see, the data transfer time continues to increase as the batch size grows, while the (de)encryption time is small and stable. This explains that the latency overhead is positively correlated with the batch size. Since the intensive computation overshadows the cost of data transfer, we observe that the proportion of the latency overhead in the training time is smaller than the one in the inference time.



(a) Inference

(b) Training

Fig. 8. HETEE latency overhead on single GPU with different batch sizes.

Scalability evaluation. The HETEE system supports the elastic allocation of accelerator resources. Multiple accelerators can be dynamically assigned to one enclave for speedup. Table III shows the scalability of HETEE. Two main conclusions can be drawn: (1) The elastic resource allocation mechanism of HETEE can significantly improve the performance of a single HETEE enclave. For most workloads, multiple GPUs achieve acceleration compared to a single GPU. In particular, the performance of 2 GPUs is 1.64x of a single GPU on average, and 4 GPUs achieve a 2.59x speedup on average; (2) HETEE does not affect scalability compared to the baseline. Taking ResNet152 as an example, the scalability of the HETEE system remains essentially the same as the baseline system.

TABLE III
HETEE INFERENCE THROUGHPUT SCALABILITY EVALUATION (NORMALIZED TO THE BASELINE)

Model	Batch size	4			8			16		
	Number of GPU	1 GPU	2 GPUs	4 GPUs	1 GPU	2 GPUs	4 GPUs	1 GPU	2 GPUs	4 GPUs
VGG16	Baseline	1.00	1.75	2.52	1.00	1.58	2.74	1.00	1.84	3.02
	HETEE	0.97	1.65	2.48	0.97	1.53	2.56	0.96	1.67	2.68
GoogLeNet	Baseline	1.00	1.37	1.60	1.00	1.47	1.76	1.00	1.29	1.72
	HETEE	0.94	1.33	1.45	0.81	1.38	1.49	0.63	1.02	1.24
ResNet50	Baseline	1.00	1.66	2.73	1.00	1.53	2.39	1.00	1.69	2.73
	HETEE	0.96	1.61	2.46	0.96	1.54	2.34	0.90	1.62	2.39
ResNet101	Baseline	1.00	1.73	2.89	1.00	1.65	2.71	1.00	1.64	2.76
	HETEE	0.98	1.63	2.82	0.98	1.59	2.58	0.97	1.56	2.70
ResNet152	Baseline	1.00	1.72	3.26	1.00	1.79	3.10	1.00	1.67	3.35
	HETEE	0.98	1.57	2.99	0.98	1.61	2.82	0.98	1.61	3.28
Average	Baseline	1.00	1.65	2.60	1.00	1.60	2.54	1.00	1.63	2.72
	HETEE	0.97	1.56	2.44	0.94	1.53	2.36	0.89	1.50	2.46

However, the absolute multiplication effect of multiple GPUs is not obvious, because 4 GPUs (with 4* PCIe x8: 32 Lanes) need to communicate with the same proxy node CPU that has only 1 PCIe port (PCIe x4) in our prototype. It thus causes traffic contention. This problem can be solved by increasing the I/O bandwidth connected to the PCIe fabric.

C. Resource utilization of FPGA

The security and isolation related functions in the SC such as the (de)encryption modules, the remote attestation and message queue scheduling modules, are implemented inside the Xilinx Zynq FPGA chip. The Vivado synthesis report shows that these functions consume 52.54% LUT and 43.70% FF programmable resources, which run at the frequency of 100 MHz. For the decryption and hardware data copy module, its average bandwidth is 267.67 MB/s and the latency is 44.83 ms. The encryption and hardware data copy module has an average bandwidth of 268.87 MB/s and a latency of 44.63ms.

VII. Security analysis

In this section, we present the security analysis on HETEE, from the perspectives of its TCB and attack surface, as well as the major protection.

TCB. The TCB includes the hardware, firmware and software components of the Security Controller (SC) [1], as follows:

• *Hardware components on the SC's FPGA device*. These components include key generation, message (de)encryption & Parsing & Dispatch engines and logic for secure boot and reset of the proxy nodes and accelerators. Also we trust the FPGA device's encrypted bitstream, whose integrity is ensured by the manufacturer's secure boot protection (see Sec. III-D).

• *Firmware on the SC's mCPU*. Also in the TCB is the firmware for isolation/enclave management (configuring the PCIe switch, secure reboot of the proxy node), and code for remote attestation on mCPU. Particularly, we implemented the isolation and enclave management module on a tailored open-source coreboot [87] to avoid using a more heavyweight Linux OS. Our implementation is only 300K bytes, compared with over 30M bytes of a Linux kernel.

• *GPU firmware*. We assume that the firmware does not include malicious code and its integrity is protected. The firmware of a modern GPU is under secure boot and read-only protection [88], [96]. For older GPU devices, the flash drives [89] that keep their firmware can be configured as read-only, by setting its write protection pin.

• *Micro Control Unit for the tamper-resistant HETEE box*. MCU is used by the HETEE box to detect physical tamper attempts and remove the sensitive FPGA bitstream, as utilized by existing tamper-resistant products (Sec. III-F).

Protection under the TCB/assumptions. HETEE components are protected or prevented from misbehaving under the TCB and assumptions, as summarized below:

• *Root of trust*. The endorsement key for trust establishment is the root of trust of a HETEE platform. It is kept as a constant in the encrypted FPGA bitstream. The integrity and confidentiality of the bitstream is protected by the FPGA synthesis tool and the dedicated circuit of the FPGA bitstream decryption engine.

• *Trust chain establishment*. Each time a HETEE platform is restarted, the FPGA first powers up and loads the encrypted bitstream file in its boot flash to the logic circuit. Then a dedicated measurement logic on the FPGA will ensure that the SC is initiated with trusted firmware and security management code (Sec. III-D).

• *Proxy CPU firmware protection*. To protect proxy firmware, the SC includes a proactive firmware verification mechanism: the SPI pin of the CPU boot flash chip is routed to the I/O pins of the FPGA, which allows the FPGA to measure the firmware in the flash chip and verify its integrity.

• *Protection from a compromised proxy node*. The proxy node is isolated from the SC and can only access its task queue. Therefore, a compromised proxy cannot tamper with the SC in the absence of security flaws. Further all HETEE enclaves (each managed by a proxy) are physically separated from each other through PCIe. During task switches, all

[1] HETEE is designed with the focus on protecting PaaS platform. Under this service model, HETEE is meant to provide the following security guarantees: an enclave user's data is protected (1) from the enclaves on other proxy nodes, (2) from past and future users of the same proxy node (3) from untrusted hosts and (4) from the cloud provider. These guarantees are achieved by the physical isolation that separates different enclaves and an enclave from hosts, secure rebooting that checks the integrity of the platform software to avoid infection from past users and chassis-level protection to detect unauthorized physical access to the HETEE box. Since none of such protection has been implemented on the proxy node, we consider the node outside the TCB.

accelerators and proxy nodes will be reset to secure states (assuming that GPU firmware is trusted, as discussed below).

- *GPU firmware protection*. Today's GPU vendors have already taken measures to protect the firmware. For example, the latest GPUs support firmware signature checking, which prevents unauthorized firmware modification [88], [96]. For some of the older generation of GPUs, the flash chip storing the firmware can be configured as read-only using its write protection (WP) pin [89] to defeat a tampering attack.
- *Defense against physical attack*. The HETEE box is protected by mature tamper-resistant techniques, and a typical solution can be found in Sec. III-F.

Comparison with other approaches. We compare our assumptions with those underlying other TEE solutions supporting GPUs, including Graviton and HIX.

- *Software stack*. In both Graviton and HIX, the GPU driver, CUDA runtime and deep learning framework run in the enclave on the CPU. They rely on the whole software stack (GPU driver, CUDA, tensorflow, etc.) for data protection, which is problematic given the large side-channel attack surface (cache, memory, CPU usage, etc.) exposed by today's CPU enclaves (SGX, TrustZone, etc.). Further, the communication between the CPU TEE and the GPU TEE might also leak information. By comparison, HETEE enclaves are physically isolated from untrusted nodes and from each other, so we do not need to trust such software to be side-channel free.
- *GPU firmware*. Like HETEE, Graviton and HIX also need to trust GPU firmware. When a GPU is assigned to a different enclave, it will also be cleaned up to prevent information leak.
- *Physical attack protection*. HIX does not work under physical attacks on PCIe interconnects and GPUs. Graviton protects against physical attack but needs to modify the GPU chip. To support unmodified GPU, HETEE adopts chassis-level protection as used in commercial HSM products.

Integrity protection. Like other TEEs (e.g., SGX, Sanctum), HETEE protects the integrity of an enclave program's execution: that is, the program's execution trace is only determined by the program's input provided by the enclave user, not by the program in a different enclave or in an untrusted host [95]; further the integrity of the enclave user's input is also protected by integrity check and authentication.

Trusted path between remote user and HETEE enclave. Remote attestation of the HETEE system uses a standard remote authentication and key agreement mechanism to support mutual authentication between the remote user and the HETEE platform, and negotiate a shared secret between them which can be further used to protect the subsequent communication. The enclave measurement report for the remote attestation includes the SC's own metrics and the HETEE enclave metrics (Sec. III-D). The shared secret is used to establish a secure channel between the user and the enclave.

Attack surfaces. Given the small attack surface (physically isolated enclave, no memory, cache, CPU sharing with the untrusted hosts and across enclaves), the adversary running on the host node or in an attack enclave can only observe the timing between an enclave's reception of its input and generation of its output, and packet size, sequence of the communication with the enclave user. Such information leak is way below that of other TEE designs. Covert channels are currently out of the scope of the paper (Sec. II-C).

VIII. Discussion

HETEE box vs. a separate server. An intuitive alternative is to deploy a separate server (e.g., standard GPU servers) in the data center as a TEE, and other nodes access its dedicated heterogeneous computing resources through an Ethernet network. Compared with a standard but separated server, HETEE is much securer and can be cheaper (given the same level of protection). First of all, our approach is characterized by the design to achieve a thin TCB. A separated server, without intensive customization, will have a thick software stack. Secondly, HETEE effectively prevents the malicious external access to the internal resources of the HETEE box through the two-level hardware strong isolation mechanism. Finally, it enables task-level isolation through a restart mechanism to prevent possible information leaks or malicious attacks between tasks. These mechanisms do not exist in the design of standard heterogeneous computing servers.

Security related cost analysis of HETEE. The cost of the HETEE box can be broken down into five major components: PCIe switch backplane, accelerators, the SC node, the proxy nodes as well as self-destructing module. As mentioned earlier, since such a pooled resource architecture is now increasingly used in the data center, PCIe switch backplane and centralized GPUs can be considered as existing resources that do not count towards the security cost of the HETEE box. Therefore, the security related cost of the HETEE box comes primarily from the SC, the proxy nodes and the self-destructing module.

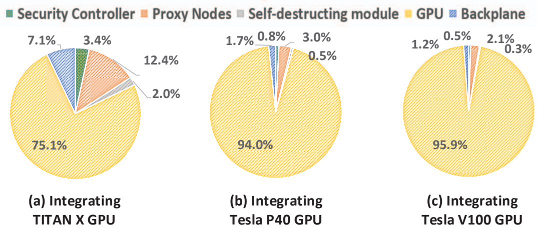


Fig. 9. HETEE Box cost analysis. (*Intel Xeon-E3 1220V6 Chip: $213 [57], Xilinx Zynq Chip:$60 [58], 16GB DDR4 2400MHz ECC: $548 [59],Self-destructing module:$500 [34], Nvidia GTX TITAN X: $1150 [60], Nvidia Tesla P40: $5999 [61], Nvidia Tesla V100: $8799 [62], Broadcom PEX9797 Chip: $590 [63];*)

A typical configuration of the HETEE box includes 1 Security Controller, 4 proxy nodes, and 16 GPUs. We choose three widely used GPUs in AI computing, and the cost breakdown is shown in Figure 9. In the case of the self-destructing module, we consider one practical solution that includes a set of pressure sensor modules, MCU control board, and disks with security protection functions. It can be seen that the security cost of the HETEE box takes about 17.8% of the total HETEE box for TITAN X GPU, and only 2.9% for the

Tesla V100 GPU. When more GPUs are integrated (e.g. 32), the portion of the security expense will go further down.

Extending HETEE to support the SaaS model. HETEE can be extended to support software-as-a-service (SaaS, in which the enclave creator also provides application software such as a genetic testing program for disease discovery [110], [111]), when the application itself does not include secret. The user can verify the integrity of the software through remote attestation, and HETEE protects the user data (e.g., DNA data) uploaded from exposing to the unauthorized party (e.g., the cloud provider who creates the enclave through HETEE).

In the case that the cloud provider's application software does contain secrets (e.g., secret parameters of a ML model) and needs to protect from the enclave user, however, an application-specific component can run on the SC for input check and sanitization [112], so as to protect the software against memory attacks from malicious inputs uploaded by the enclave user (which can lead to the exposure of the software's secret). When the application software itself is unknown to the enclave user and therefore cannot be audited through integrity check, a sandbox (similar to Ryoan [113]) could be used to prevent the unauthorized leak of user data. In this case, however, we needs to include the sandbox in the TCB.

IX. Related Works

Isolated Execution. Mainstream processor vendors have implemented TEEs in some of their chip products, such as Intel Software Guard Extensions (SGX) [3], AMD Secure Encrypted Virtualization (SEV) [4] and ARM TrustZone [5]. In addition, based on the concept of open-source security, Keystone [64] and Sanctum [65]–[67] are proposed for RISC-V processor. These TEEs generally isolate a secure world from the insecure one, and the protected data can be processed in such secure world. However, none of those TEEs can truly support CDI computing tasks which widely adopt a heterogeneous architecture. For example, Intel SGX does not support trusted I/O paths to protect the data transmissions between enclaves and I/O devices. Although ARM TrustZone can support trusted I/O paths for certain peripherals in the ARM ecosystem, it is noted that TrustZone still does not truly support heterogeneous computing units like GPU.

Trusted paths. Graviton [15] modifies the existing GPU chips via enhancing the internal hardware command processor, to support trusted execution environments on GPUs. HIX [16] extends an SGX-like design to enable secure access to GPU from the CPU enclave, which needs to modify the MMU and PCIe Root Complex on the CPU chip. By comparison, HETEE does not require any changes to existing commercial CPUs or accelerators. SGXIO [68] is a generic trusted path extension for Intel SGX. Trusted paths are established via a dedicated trusted boot enclave. However, the capacity limitation of the SGX enclave prevent SGXIO being widely used for high-performance accelerators like off-chip GPUs. Besides, there are several prior studies that propose specific trusted paths for some extertnal devices but not PCIe accelerators. For instance, Zhou et al. propose a trusted path to a single USB device [69]. Yu et al. demonstrate how to build a trusted path for display [70]. Filyanov et al. discuss a pure uni-directional trusted path using the TPM and Intel TXT [71].

Privacy preserving deep learning. Nick Hynes et al evaluated two types of secure AI computing [72]. One scenario is to compute the entire AI workloads inside SGX enclaves, which cannot utilize accelerators. Similar work can be found in other studies [83], [84]. Another scenario is the Slalom solution [73]. It needs to decompose the AI model network into two parts, in which the upper control flow part runs inside the SGX enclave and is tightly protected, while some non-privacy-sensitive basic computations is thrown to the untrusted GPU for acceleration. However, splitting AI networks result in possible accuracy decrease, while our HETEE programming model securely encapsulate the whole AI network, without any change of the AI structure, so the accuracy will not be affected. Researchers also proposed to protect the privacy of AI networks by introducing noise [85].

X. Conclusion

Large-scale confidential computing is driven by huge demands in the real world. However, it still cannot be supported by today's Trusted Execution Environment (TEE), due to the lack of scalable and effective protection of high-throughput accelerators like GPUs. To address these problems, this paper presents the first Heterogeneous TEE design that can truly support large-scale compute and/or data intensive (CDI) computing, without any chip-level change. Our approach, called *HETEE*, is a device for centralized management of all computing units of a server rack. It is uniquely designed to work with today's data centers and clouds, leveraging modern resource pooling technologies to dynamically compartmentalize computing tasks, and enforce strong isolation and reduce TCB through hardware support. More specifically, HETEE utilizes the PCIe switch fabric to allocate its accelerators to the server node on the same rack for a non-sensitive CDI task, and move them back into a secure enclave in response to the demand for confidential computing. We implemented HETEE on a real hardware system, and evaluated it with popular neural network inference and training tasks. Our evaluations show that HETEE can easily support such rack-scale confidential computing tasks with a small performance overhead and exhibits good scalability when elastically using multiple accelerators. Down the road, we plan to further evaluate our design and implementation through formal verification, and investigate hardware-software separation of the design.

XI. Acknowledgments

We thank the anonymous reviewers and the shepherd Prof. Emmett Witchel for their insightful comments and suggestions. We also thank Dr. Shijun Zhao for his help in remote attestation protocol. This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences under grant No. XDC02010200 and National Natural Science Foundation of China (Grant No.61802397).

REFERENCES

[1] Josh Constine. Facebook bug exposed up to 6.8M users' unposted photos to apps, https://techcrunch.com/2018/12/14/facebook-photobug/, 2018.

[2] CBSNEWS. Facebook CEO warns data exposure could be more widespread, https://www.cbsnews.com/video/facebook-ceo-warns-data-exposure-could-be-more-widespread/, 2018.

[3] Intel. Intel Software Guard Extensions (Intel SGX), https://software.intel.com/en-us/sgx/details, 2019.

[4] AMD. AMD Secure Encrypted Virtualization (SEV), https://developer.amd.com/sev/, 2019.

[5] ARM. Architecting a more Secure world with isolation and virtualization, https://developer.arm.com/products/architecture/security-architectures?_ga=2.65596206.1465614028.1550155414-1474563975.1550155414, 2019.

[6] Facebooks. Accelerating Facebook's infrastructure with application-specific hardware. POSTED ON MAR 14, 2019 TO DATA CENTER ENGINEERING. By Kevin Lee, Vijay Rao, William Christie Arnold. https://code.fb.com/data-center-engineering/accelerating-infrastructure/

[7] Amazon. Machine Learning on AWS: Putting machine learning in the hands of every developer, https://aws.amazon.com/machinelearning/?hp=tile&tile=solutions, 2019.

[8] Adrian M Caulfield, Eric S Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Jooyoung Kim, et al. A cloud-scale acceleration architecture. international symposium on microarchitecture, pages 1-13, 2016.

[9] Jeremy Fowers, Jooyoung Kim, Doug Burger, and Scott Hauck. A scalable high-bandwidth architecture for lossless compression on fpgas. pages 52-59, 2015.

[10] Google. Cloud Tensor Processing Units (TPUs), https://cloud.google.com/tpu/, 2019.

[11] Muhuan Huang, Di Wu, Cody Hao Yu, Zhenman Fang, Matteo Interlandi, Tyson Condie, and Jason Cong. Programming and runtime support to blaze fpga accelerator deployment at datacenter scale. In Proceedings of the Seventh ACM Symposium on Cloud Computing, pages 456-469. ACM, 2016.

[12] Microsoft. Azure Reference Architectures, https://docs.microsoft.com/enus/azure/architecture/reference-architectures/, 2019.

[13] Andrew Putnam, Adrian M Caulfield, Eric S Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, et al. A reconfigurable fabric for accelerating large-scale datacenter services. international symposium on computer architecture, 42(3):13-24, 2014.

[14] Charles Roe. The Future of the Data Center: Heterogeneous Computing, https://www.dataversity.net/future-data-center-heterogeneous-computing/, 2016.

[15] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. Graviton: Trusted execution environments on gpus. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pages 681-696, Carlsbad, CA, 2018. USENIX Association.

[16] Jang, Insu and Tang, Adrian and Kim, Taehoon and Sethumadhavan, Simha and Huh, Jaehyuk. Heterogeneous Isolated Execution for Commodity GPUs. Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '19, pages 455-468. ACM, 2019

[17] NVIDIA. CUDA Toolkit Documentation v9.0.176. https://docs.nvidia.com/cuda/archive/9.0/

[18] Google. TensorFlow API documents, reversion 1.11. https://tensorflow.google.cn/versions/r1.11/api_docs/python/tf

[19] NVIDIA DGX-2. NVSwitch Accelerates NVIDIA DGX-2. By Robert Sohigian. August 21, 2018. https://devblogs.nvidia.com/nvswitch-accelerates-nvidia-dgx2/

[20] NVIDIA DGX-2 Details at Hot Chips 30. By Patrick Kennedy. August 21, 2018. https://www.servethehome.com/nvidia-dgx-2-details-at-hot-chips-30/

[21] NVIDIA HGX-2. WORLD'S MOST POWERFUL ACCELERATED SERVER PLATFORM FOR DEEP LEARNING, MACHINE LEARNING, AND HPC. https://www.nvidia.com/en-us/data-center/hgx/

[22] NVIDIA. HGX-2 Fuses HPC and AI Computing Architectures. By William. May 29, 2018. https://devblogs.nvidia.com/hgx-2-fuses-ai-computing/

[23] HGX-1. Siamak Tavallaei, CSI, Azure Cloud. Microsoft Project Olympus Hyperscale GPU Accelerator. May 26, 2017. https://azure.microsoft.com/mediahandler/files/resourcefiles/00c18868-eba9-43d5-b8c6-e59f9fa219ee/HGX-1%20Blog_5_26_2017.pdf

[24] Siamak Tavallaei, Robert Ober.Microsoft Project Olympus Hyperscale GPU Accelerator (HGX-1). OCP U.S. SUMMIT 2017. Santa Clara, CA, March 8, 2017. http://schd.ws/hosted_files/ocpussummit2017/85/OCP17%20Microsoft%20Project%20Olympus%20Hyperscale%20GPU%20Accelerator_HGX-1_March_8_2017.pdf

[25] Broadcom. PEX9797. 97 lane, 25 port, PCI Express Gen3 ExpressFabric Platform. https://www.broadcom.com/products/pcie-switches-bridges/expressfabric/pex9797

[26] Avago's PEX9700 turns the PLX PCIe3 switch into a fabric. May 12, 2015 by Charlie Demerjian. https://semiaccurate.com/2015/05/12/avagos-pex9700-turns-plx-pcie3-switch-fabric/

[27] Avago Announces PLX PEX9700 Series PCIe Switches: Focusing on Data Center and Racks. by Ian Cutress on May 12, 2015. https://www.anandtech.com/show/9245/avago-announces-plx-pex9700-series-pcie-switches

[28] Peter X. Gao and Akshay Narayan and Sagar Karandikar and Joao Carreira and Sangjin Han and Rachit Agarwal and Sylvia Ratnasamy and Scott Shenker. Network Requirements for Resource Disaggregation. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pages 249-264, Savannah, GA, 2016. USENIX Association. https://www.usenix.org/system/files/conference/osdi16/osdi16-gao.pdf

[29] Huawei. High Throughput Computing Data Center Architecture, Thinking of Data Center 3.0. Technical White Paper. June, 2014. https://www.huawei.com/ilink/en/download/HW_349607

[30] EDP Europe. Infrasolution rack access control. https://www.edpeurope.com/product/rack-accesscontrol/.

[31] EDP Europe. iaccess it cabinet security system. https://www.edpeurope.com/product/iaccesscomputer-cabinet-door-security-access-controlsystem/.

[32] EDP Europe. Biometric swipe access card. https://www.edpeurope.com/product/biometric-swipe-access-card.

[33] Southco. Rack-level security. http://lp.southco.com/rs/southco/images/Rack%20Level%20Security%20Brochure.pdf.

[34] Gary Smolker and Leon Chernick. Method and apparatus for combustibly destroying microelectronic circuit board interconnections, May 6 1975. US Patent 3,882,324.

[35] Open Neural Network Exchange Format, The open ecosystem for interchangeable AI models. https://onnx.ai

[36] Open Neural Network Exchange (ONNX). Tutorials for creating and using ONNX models. https://github.com/onnx/tutorials

[37] Dean, Jeffrey and Barroso, Luiz Andre. The tail at scale. Communications of The ACM 2013. pages 74-80.

[38] Cong, Jason and Ghodrat, Mohammad Ali and Gill, Michael and Grigorian, Beayna and Gururaj, Karthik and Reinman, Glenn. Accelerator-Rich Architectures: Opportunities and Progresses. pages 1-6, 2014.

[39] Barroso, Luiz Andre and Clidaras, Jimmy and Holzle, Urs. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second edition. Synthesis Lectures on Computer Architecture, pages 1-154, 2013.

[40] Chen, Quan and Yang, Hailong and Mars, Jason and Tang, Lingjia. Baymax: QoS Awareness and Increased Utilization for Non-Preemptive Accelerators in Warehouse Scale Computers. Operating Systems Review 2016, pages 681-696.

[41] Jain, Paras and Mo, Xiangxi and Jain, Ajay and Subbaraj, Harikaran and Durrani, Rehan Sohail and Tumanov, Alexey and Gonzalez, Joseph E and Stoica, Ion. Dynamic Space-Time Scheduling for GPU Inference. arXiv: Distributed, Parallel, and Cluster Computing, 2019. https://arxiv.org/pdf/1901.00041.pdf

[42] Zhichao Hua and Jinyu Gu and Yubin Xia and Haibo Chen and Binyu Zang and Haibing Guan. vTZ: Virtualizing ARM TrustZone. 26th USENIX Security Symposium (USENIX Security 17), pages 541-556, Vancouver, BC, 2017. USENIX Association. https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-hua.pdf

[43] Caffe. Yangqing Jia. http://caffe.berkeleyvision.org

[44] Caffe: a fast open framework for deep learning. https://github.com/BVLC/caffe/

[45] PyTorch. FROM RESEARCH TO PRODUCTION. An open source deep learning platform that provides a seamless path from research prototyping to production deployment. https://pytorch.org

[46] PyTorch. Tensors and Dynamic neural networks in Python with strong GPU acceleration. https://github.com/pytorch/pytorch

[47] PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology, Revision 2.5, January 2011. https://www.intel.com/content/dam/doc/application-note/pci-sig-sr-iov-primer-sr-iov-technology-paper.pdf

[48] SR-IOV Architecture. 04/20/2017, Duncan MacMichael. https://docs.microsoft.com/en-us/windows-hardware/drivers/network/sr-iov-architecture

[49] Michael Krause (HP, co-chair), Renato Recio (IBM, co-chair). PCIe I/O virtualization and sharing. PCI-SIG 2006.
http://weblab.cs.uml.edu/ bill/cs520/slides_15D_PCI_Express_IOV.pdf
http://weblab.cs.uml.edu/ bill/cs520/slides_15E_PCI_Express_IOV.pdf

[50] Using PCI express as the primary system interconnect in multiroot compute storage communications and embedded systems (White Paper). 2008. https://www.idt.com/document/whp/idt-pcie-multi-root-white-paper

[51] SGD. Stochastic gradient descent. https://en.wikipedia.org/wiki/Stochastic_gradient_descent

[52] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations (ICLR2015), San Diego, 2015. https://arxiv.org/abs/1412.6980

[53] ImageNet. Large Scale Visual Recognition Challenge, 2012 (ILSVRC2012). http://image-net.org/challenges/LSVRC/2012/

[54] VGG16. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. Computer Science, 2014.

[55] GoogLeNet. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2014.

[56] ResNet. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learningfor image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 00, pages 770-778, June 2016.

[57] Intel. INTEL XEON PROCESSOR E3-1220 V6. https://www.intel.com/content/www/us/en/products/processors/xeon/e3-processors/e3-1220-v6.html

[58] Xilinx. Xilinx Zynq-7000, XC7Z010-1CLG400C. https://www.digikey.com/product-detail/en/xilinx-inc/XC7Z010-1CLG400C/122-1854-ND/3925789?_ga=2.250576899.442891328.1561903744-1937502679.1561903744

[59] Dell. Dell memory 2RX8 DDR4 UDIMM. https://www.dell.com/en-us/shop/accessories/apd/a9755388

[60] Amazon. GeForce GTX TITAN X. https://www.amazon.com/EVGA-GeForce-GAMING-Graphics-12G-P4-2990-KR/dp/B00UVN21RQ

[61] THINKMATE. NVIDIA Tesla P40, https://www.thinkmate.com/hardware/co-processors

[62] THINKMATE. NVIDIA Tesla V100, https://www.thinkmate.com/hardware/co-processors

[63] Broadcom. PCIe Switch Chip Data Sheet, https://www.digikey.com/product-detail/en/broadcomlimited/PEX9797-AA80BC-G/516-3574-ND/6139515

[64] Dayeol Lee, Dawn Song, Llia Lebedev, Srini Devadas, and etc. Keystone: Open-source secure hardware enclave. https://keystone-enclave.org/

[65] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In 25th USENIX Security Symposium (USENIX Security 16), pages 857-874, Austin, TX, 2016. USENIX Association.

[66] Ilia Lebedev, Kyle Hogan, and Srinivas Devadas. Secure boot and remote attestation in the sanctum processor. Cryptology ePrint Archive, Report 2018/427, 2018. https://eprint.iacr.org/2018/427.

[67] Ilia Lebedev and Kyle Hogan and Jules Drean and David Kohlbrenner and Dayeol Lee and Krste Asanović and Dawn Song and Srinivas Devadas. Sanctorum: A lightweight security monitor for secure enclaves. Cryptology ePrint Archive, Report 2019/001. https://eprint.iacr.org/2019/001

[68] Samuel Weiser and Mario Werner. Sgxio: Generic trusted i/o path for intel sgx. In Proceedings of the Seventh ACM Conference on Data and Application Security and Privacy, pages 261-268, 2017.

[69] Z. Zhou, M. Yu, and V. D. Gligor. Dancing with giants: Wimpy kernels for on-demand isolated i/o. In 2014 IEEE Symposium on Security and Privacy, pages 308-323, May 2014.

[70] Miao Yu, Virgil D. Gligor, and Zongwei Zhou. Trusted display on untrusted commodity platforms. In Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, pages 989-1003, New York, NY, USA, 2015. ACM.

[71] A. Filyanov, J. M. McCuney, A. Sadeghiz, and M. Winandy. Unidirectional trusted path: Transaction confirmation on just one device. In 2011 IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN), pages 1-12, June 2011.

[72] Nick Hynes, Raymond Cheng, and Dawn Song. Efficient deep learning on multi-source private data, CoRR 2018. Aug 13, 2018. https://arxiv.org/abs/1807.06689.

[73] Florian Tramer and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware, 2018. https://arxiv.org/abs/1806.03287.

[74] Mohan J. Kumar. Rack Scale Architecture for Cloud[EB/OL]. https://packetpushers.net/wp-content/uploads/2014/09/Rack-Scale-Architecture-%E2%80%93-Platform-and-Management-SF14_DATS008_104f.pdf :Intel IDF, 2013.

[75] Putnam A, Caulfield A M, Chung E S, et al. A reconfigurable fabric for accelerating large-scale datacenter services[J]. IEEE Micro, 2015, 35(3): 10-22..

[76] Lim K T, Chang J, Mudge T N, et al. Disaggregated memory for expansion and sharing in blade servers[J]. international symposium on computer architecture, 2009, 37(3): 267-278.

[77] Novakovic S, Daglis A, Bugnion E, et al. Scale-out NUMA[J]. architectural support for programming languages and operating systems, 2014, 49(4): 3-18.

[78] Rui Hou, Tao Jiang, Liuhang Zhang, Pengfei Qi, Jianbo Dong, Haibin Wang, Xiongli Gu, and Shujie Zhang, Cost Effective Data Center Servers[C], In:Proceedings of the 19th IEEE International Symposium on High Performance Computer Architecture (HPCA),Shenzhen,China, 2013, pp179-187.

[79] Jianbo Dong, Rui Hou, Michael Huang, Tao Jiang, Boyan Zhao, Sally A.McKee, Haibin Wang, Xiaosong Cui, and Lixin Zhang Venice: Exploring server architectures for effective resource sharing,[C] in High Performance Computer Architecture (HPCA),Barcelona, Spain,2016

[80] Hoda Naghibijouybari, Ajaya Neupane, Zhiyun Qian, and Nael Abu-Ghazaleh. Rendered insecure: Gpu side channel attacks are practical. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 2139–2153. ACM, 2018.

[81] Zhenhang Jiang, Yunsi Fei, and David Kaeli. A novel side-channel timing attack on gpus. In Proceedings of the on Great Lakes Symposium on VLSI 2017, pages 167–172. ACM, 2017.

[82] Roberto Di Pietro, Flavio Lombardi, and Antonio Villani. Cuda leaks: a detailed hack for cuda and a (partial) fix. ACM Transactions on Embedded Computing Systems (TECS), 15(1):15, 2016.

[83] HESAMIFARD E, TAKABI H, GHASEMI M, et al. Privacy-preserving machine learning in cloud[C]//The 2017 on Cloud Computing Security Workshop. 2017: 39-43.

[84] Gu Z, Huang H , Zhang J , et al. Securing Input Data of Deep Learning Inference Systems via Partitioned Enclave Execution[J]. 2018.

[85] DWORK C, MCSHERRY F, NISSIM K, et al. Calibrating noise to sensitivity in private data analysis[C]//The Third conference on Theory of Cryptography. 2006: 265-284.

[86] Kyle Wilkinson, Using Encryption to Secure a 7 Series FPGA Bitstream. XAPP1239 (v1.1) July 16, 2018. https://www.xilinx.com/support/documentation/application_notes/xapp1239-fpga-bitstream-encryption.pdf

[87] J. Sun, M. Jones, S. Reinauer, and V. Zimmer, Building coreboot with Intel FSP, pp. 55–95. Berkeley, CA: Apress, 2015. https://link.springer.com/chapter/10.1007/978-1-4842-0070-4_4

[88] NVIDIA Sends Out Signed Firmware Images For GP108 Pascal GPUs. Written by Michael Larabel in NVIDIA, on 21 December 2017. https://www.phoronix.com/scan.php?page=news_item&px=NVIDIA-GP108-Firmware

[89] Winbond SPI Flash W25Q64FV. 3V 64M-BIT SERIAL FLASH MEMORY WITH DUAL/QUAD SPI & QPI. Publication Release Date: July 18, 2017. Revision S. https://4donline.ihs.com/images/VipMasterIC/IC/WBND/WBND-S-A0003377185/WBND-S-A0003377185-1.pdf

[90] H. Krawczyk, "Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike protocols," inAdvances in Cryptology - CRYPTO 2003(D. Boneh, ed.), (Berlin, Heidelberg), pp. 400–425, Springer Berlin Heidelberg,2003.

[91] Jayesh Vyas Lavina Jain. Security Analysis of Remote Attestation,https://seclab.stanford.edu/pcl/cs259/projects/cs259_final_lavina_jayesh/CS259_report_lavina_jayesh.pdf, 2008.

[92] V. Costan and S. Devadas, "Intel SGX Explained,"IACR Cryptology ePrint Archive, vol. 2016, p. 86, 2016.

[93] Intel SGX: EPID Provisioning and Attestation Services: https://software.intel.com/sites/default/files/managed/57/0e/ww10-2016-sgx-provisioning-and-attestation-final.pdf

[94] Intel Software Guard Extensions Remote Attestation End-to-End Example, July 4, 2018: https://software.intel.com/en-us/articles/code-sample-intel-software-guard-extensions-remote-attestation-end-to-end-example

[95] Pramod Subramanyan, Rohit Sinha, Ilia Lebedev, Srinivas Devadas, and Sanjit A. Seshia. A formal foundation for secureremote execution of enclaves. InProceedings of the 2017 ACM SIGSAC Conference on Computer and CommunicationsSecurity, CCS '17, pages 2435–2450, New York, NY, USA, 2017. ACM.

[96] torpcoms. AMD Vega GPU Firmware Signing, https://forum.level1techs.com/t/amd-vega-firmware-signing/118459, 2017.

[97] Private Machines. ENFORCER SRX1 Tamper-Proof Server and ENFORCER R1 HSM blade,https://privatemachines.com/enforcer/, 2019.

[98] S. F. Yitbarek, M. T. Aga, R. Das, and T. Austin. Cold boot attacks are still hot: Security analysis of memory scramblers in modern processors. In2017 IEEE International Symposium on High Performance Computer Architecture (HPCA),pages 313–324, Feb 2017.

[99] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J.Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In17thUSENIX Security Symposium (USENIX Security 08), San Jose, CA, July 2008. USENIX Association.

[100] M. Gruhn and T. Muller. On the practicability of cold boot attacks. In 2013 International Conference on Availability,Reliability and Security, pages 390–397, Sep. 2013.

[101] Shashank Pandey, Niladri Banerjee, Yan Xie, and C.H. Mastrangelo. Self-destructing secured microchips by on-chip triggered energetic and corrosive attacks for transient electronics.Advanced Materials Technologies, 3:1800044, 06 2018.

[102] Poluka Srilatha. Self Destructing Electronics, http://www.iosrjournals.org/iosr-jeee/Papers/Vol9-issue3/Version-6/D09362430.pdf, May 2014.

[103] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a tcg-based integritymeasurement architecture. In Proceedings of the 13th USENIX Security Symposium, pages 223–238, 01 2004.

[104] Eric Peeters, Bhargavi Nisarga. Texas Instruments: System-Level Tamper Protection Using MSP MCUs, https://privatemachines.com/enforcer/, August 2016.

[105] Daniel Nelson, etc. ORWL: an open source, physically Secure Microcontroller, https://www.crowdsupply.com/design-shift/orwl/updates/secure-microcontroller-details, September 2016.

[106] Information Technology Laboratory National Institute of Standards and Technology. NIST.FIPS.140-2 SECURITYREQUIREMENTS FOR CRYPTOGRAPHIC MODULES,https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf, January 1994.

[107] National Institute of Standards and Technology Canadian Centre for Cyber Security. FIPS1402IG-Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program,https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402IG.pdf, October 2019.

[108] IBM Cloud Hardware Security Module 7.0, Secure key storage and cryptographic operations within a FIPS 140-2 Level 3,https://www.ibm.com/cloud/hardware-security-module, 2019.

[109] Microsoft Azure Mark Russinovich Chief Technology Officer. Azure confidential computing, https://azure.microsoft.com/en-us/blog/azure-confidential-computing/, 2018.

[110] Microsoft. Scalable and Secure Genomic Data Analysis on Azure, https://www.microsoft.com/en-us/genomics/, 2019.

[111] google cloud. life-sciences, https://cloud.google.com/life-sciences/, 2019.

[112] Robert Seacord (Manager). Input Validation and Data Sanitization, https://wiki.sei.cmu.edu/confluence/display/java/Input+Validation+and+Data+Sanitization, 2015.

[113] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. Ryoan: A distributed sandbox for untrustedcomputation on secret data. In12th USENIX Symposium on Operatin

[114] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Dawn Song, and Krste Asanovic. Keystone: A framework for architecting tees. CoRR, abs/1907.10119, 2019.

[115] Liqun Chen and Bogdan Warinschi. Security of the TCG Privacy-CA Solution, https://www.researchgate.net/publication/221452123_Security_of_the_TCG_Privacy-CA_Solution. In Conference: IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing, EUC 2010, Hong Kong, China, 11-13 December 2010, pages 609-616, 12 2010.

[116] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS'04, page 132-145, New York, NY, USA, 2004. Association for Computing Machinery.

[117] Shijun Zhao, Qianying Zhang, Yu Qin, Wei Feng, and Dengguo Feng. Sectee: A software-based approach to secure enclave architecture using tee. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS'19, page 1723-1740, New York, NY, USA, 2019. Association for Computing Machinery.

[118] Bryan Parno. Bootstrapping Trust in a "Trusted" Platform. https://www.usenix.org/legacy/event/hotsec08/tech/full_papers/parno/parno.pdf. In Conference: 3rd USENIX Workshop on Hot Topics in Security, July 29, 2008, San Jose, CA, USA, Proceedings, page 6, 07 2008.

[119] Aritra Dhar, Ivan Puddu, Kari Kostiainen, and Srdjan Capkun. Proximitee: Hardened sgx attestation and trusted path through proximity verification. IACR Cryptology ePrint Archive, 2018:902, 2018.

[120] Russell A. Fink, Alan T. Sherman, Alexander O. Mitchell, and David C. Challener. Catching the cuckoo: Verifying tpm proximity using a quote timing side-channel. In Trust and Trustworthy Computing (TRUST), 2011.