INVITED: AI Utopia or Dystopia - On Securing AI Platforms

Ghada Dessouky
ghada.dessouky@trust.tu-darmstadt.de
Technical University of Darmstadt, Germany

Patrick Jauernig
patrick.jauernig@trust.tu-darmstadt.de
Technical University of Darmstadt, Germany

Nele Mentens
nele.mentens@kuleuven.be
KU Leuven, Belgium

Ahmad-Reza Sadeghi
ahmad.sadeghi@trust.tu-darmstadt.de
Technical University of Darmstadt, Germany

Emmanuel Stapf
emmanuel.stapf@trust.tu-darmstadt.de
Technical University of Darmstadt, Germany

ABSTRACT

Today we are witnessing the widespread deployment of AI algorithms on many computing platforms already to provide various services, thus driving the growing market for AI-based platforms. On the one end, AI support is demanded for resource-constrained embedded devices, e.g., integrated into smart homes and vehicles. On the other end, hi-tech giants and cloud services require AI platforms with increasing computational power to feed their data-hungry neural networks. Neglecting security and privacy aspects on both such low-end and high-end AI platforms can have devastating consequences for end users (privacy and safety) as well as for the AI service providers (IP theft). The utopia of a world where intelligent devices ease the human life can easily turn into a dystopia where the ownership of personal data is threatened.

In recent years, tremendous effort has been invested in the development of security architectures that protect sensitive services in isolated execution contexts, called *enclaves*, which provide protection beyond that of commodity operating systems. In this paper, we elaborate on the most well-known enclave-based security architectures to protect AI services. We point out their shortcomings in providing the security guarantees needed for existing and emerging AI services and discuss new ideas and research directions.

1 INTRODUCTION

Artificial Intelligence (AI) is one of the current megatrends in industry and society [42]. While AI started as a supporting technology, it became a key technology that replaced various standard practices established within the last decades. Nowadays, AI is fully integrated into industrial processes: a recent survey in the UK [43] shows that every second retailer questioned uses AI, e.g., for chatbots, but also sensitive tasks like data analytics, warehousing, or finance. However, as with many rapidly emerging technologies, e.g., the Internet of Things (IoT), safety and security are only considered after widespread adoption, leaving AI's future on the edge between an AI-based utopia of self-driving cars and personal robot assistants, and a dystopian nightmare of privacy infringements and malicious surgical strikes that exploit AI weaknesses to shut down companies and whole societies.

A large number of our everyday devices, e.g., mobile and handheld devices or IoT devices like Amazon Echo or Google Home, leverage AI algorithms to provide services like speech or image processing. AI algorithms are even found in cars, enabling the biggest hype in the automotive industry in the last years, namely autonomous driving. One of the common characteristics of all these use cases is that the processed data is collected by the devices themselves using integrated sensors, e.g., microphones, cameras, fingerprint sensors, or ultrasonic sensors. This data can be privacy- or safety-critical, and hence, needs to remain confidential and integrity-protected to prevent privacy violations or compromised systems.

Nowadays, the AI algorithms and processed data sets often reach a complexity that prevents a satisfiable performance when only using the main CPU for the AI computations. Hence, on most platforms, from high-performance clusters down to mobile devices, AI algorithms utilize hardware accelerators such as Graphics Processing Units (GPU) or Neural Processing Units (NPU), as their highly parallel computations yield significantly higher AI performance [9, 19]. NPUs are widely used across different vendors, e.g., TPU by Google, Ascend by Huawei, or Apple's Neural Engine.

In this paper, we explore an important aspect which is crucial for a successful AI-based future: how to guarantee security for AI services which includes keeping the AI models confidential in terms of IP protection, as well as controlling access to the user data to prevent misuse. Moreover, the protection mechanisms must also cover usage scenarios in which the AI algorithms are either computed on hardware accelerators or in which the user data is collected on the device using integrated sensors.

A promising approach to protect AI services are enclave-based security architectures, also called Trusted Execution Environments. Enclave-based security architectures have been an appealing research subject for a wide range of computer systems, from low-end embedded devices to powerful cloud servers. The goal of these architectures is to protect sensitive services in isolated execution contexts, called *enclaves*. We survey popular enclave-based security architectures deployed for protecting AI services, compare them with regards to their suitability for AI computations and present an outlook on future research directions.

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

2 PROTECTING AI SERVICES

In the last years, the widespread deployment of AI services on today's computing devices stimulated research on the protection of sensitive AI services. Since many services process privacy- and security-critical data, most of the proposed solutions utilize or extend enclave-based security architectures which promise a high level of protection. In this section, we briefly introduce the enclave-based security architectures used so far for protecting AI services and their usage scenarios. In particular, we distinguish between solutions which run the AI computations on the main CPU and those which offload them to accelerators, e.g. GPUs, which are commonly used to boost the performance of AI algorithms.

2.1 Enclave-based Security Architectures

Enclaves on x86. Intel's Software Guard Extensions (SGX) [13], introduced in 2015, provide enclaves in user space on Intel CPUs. Modifications at the page table walker and new SGX microcode allow to protect sensitive services inside an enclave from malicious accesses from the OS and also the hypervisor. However, SGX enclaves still rely on the OS for memory management, interrupt handling and I/O services. The newly introduced Memory Encryption Engine (MEE) takes care of the encryption and integrity protection of all enclave code and data stored in the main memory.

In 2016, AMD introduced an own security architecture for their x86 processors called Secure Encrypted Virtualization (SEV) [7], which targets x86 servers and provides enclaves that comprise an entire virtual machine (VM). SEV relies on the Secure Processor (SP), introduced in AMD's Secure Memory Encryption (SME) [7] technology, to encrypt all VM code and data in the main memory with individually-generated encryption keys, which are only accessible to the SP hardware. Thus, SEV protects the VMs from a compromised hypervisor. However, the VMs still rely on the hypervisor for the management of memory and peripherals. Two extensions to SEV were so far introduced by AMD. In 2017, SEV-ES (Encrypted State) [41] was introduced, which includes the CPU registers into the protected VM state. With SEV-SNP (Secure Nested Paging) [2], introduced in 2020, the VM state is now also integrity-protected. A more detailed comparison of SGX and SEV can be found in [24].

Enclaves on ARM. ARM TrustZone, introduced in 2004 [1], separates the system into two worlds, the *normal* and the *secure* world, by introducing security enhancements at the CPU, system bus and additional components on the SoC, e.g., the memory controller. The normal world contains the commodity untrusted OS and all non-sensitive services, whereas all sensitive services run in the secure world together with an own trusted OS which performs memory management, interrupt handling and I/O tasks exclusively for the sensitive services. The main limitation of TrustZone is that all code running in the secure world needs to be trusted and thus, TrustZone can only provide a single enclave.

Sanctuary [12] overcomes the main limitations of TrustZone and provides an arbitrary number of enclaves on ARM platforms. Sanctuary does not introduce new hardware components but exploits existing TrustZone functionalities to protect sensitive services in the normal world on temporarily isolated physical CPU cores.

Enclaves on RISC-V. On the open RISC-V platform, Keystone [8] utilizes the Physical Memory Protection (PMP) unit, specified in the RISC-V privileged ISA [39] to provide enclaves. In contrast to other RISC-V based enclave architectures [6], Keystone provides enclaves which comprise of the user and supervisor software layers. Each Keystone enclave contains a runtime which can perform memory management and interrupt handling tasks for the enclave. The configuration of the PMP unit is performed by a trusted software component, called Security Monitor (SM), which runs in the highest privileged software layer of a RISC-V processor, the machine level.

2.2 AI Computing on the CPU

Protecting customer data and AI models. In most of the works on AI service protection, SGX is utilized as the underlying enclave-based security architecture. Commonly, an AI as a Service (AIaaS) scenario is assumed, whereas the proposed solutions aim at protecting the proprietary AI models of the AI service provider or the privacy-sensitive data of the AI customer.

Ohrimenko et al. [23] also assume an AIaaS scenario in which sensitive data from multiple mutually distrusted AI customers are aggregated on a remote server to train an AI model inside of an SGX enclave. Thus, the privacy-sensitive data are protected from the AIaaS provider and also other AI customers. After the training, the models are shared with all AI customers to perform inference upon them. A similar setup is assumed in Chiron [10]. However, in Chiron, the model training is performed in a Ryoan [31] sandbox (based on SGX), which still offers the AIaaS provider the possibility to freely select, configure and train the AI models. In MlCapsule [17], an AIaaS scenario is assumed as well. However, in contrast to Chiron and the work from Ohrimenko et al., the trained AI model used for inference is protected on the AI customer site inside an SGX enclave, thereby supporting an offline AIaaS scenario. VoiceGuard [11] focuses on the use case of speech processing. In VoiceGuard, sensitive audio data are collected from user devices, e.g., smart home devices, and sent to an AI service provider where the data are used for inference on proprietary AI models. Again, SGX enclaves are used to shield the inference computations to protect the user data.

Other works aim at making the protection of AI services more practical. TensorSCONE [25] introduces a general AI framework for SGX enclaves based on Tensorflow [19]. Occlumency [32] boosts the performance of deep learning inference when performed in an SGX enclave. The authors implement Occlumency using the Caffe deep learning framework [33].

All related work mentioned so far makes use of SGX enclaves. A general performance comparison between the use of SGX and SEV enclaves for AI protection was conducted by Akram et al. [4] and Göttel et al. [5]. For RISC-V based platforms, the authors of Keystone [8] showcase how AI models can be protected in a Keystone enclave when performing model learning or inference.

Secure data collection. None of the described proposals discuss how privacy-sensitive user data can be securely collected on a user device, which is especially critical in the case of biometric data. SGX, which is mostly used as the dedicated enclave-based security architecture, does not provide secure communication channels from enclaves to peripherals that are not able to encrypt their data streams, e.g., commodity biometric sensors, microphones or cameras. Thus, the user data is endangered since it could be copied by malicious software running on the user device before it is isolated

in an enclave. On ARM-based devices, the problem of secure data collection can be solved by leveraging TrustZone's capabilities to assign peripherals exclusively to the secure world. In fact, Google demands that Android devices make the raw biometric sensor data only accessible to the enclave-based security architecture (or a dedicated security chip if available) [40]. Unfortunately, TrustZone's described limitations make it necessary to cluster all critical sensor drivers and access permissions in one single enclave.
Offline Model Guard (OMG) [27], which is based on Sanctuary, provides protection for AI services on ARM-based devices without being limited to a single enclave, which is shown for the use case of speech recognition. Thus, OMG allows to protect the proprietary AI models of the AI provider and additionally, the privacy-sensitive user data, from the point of collection to the point of processing.

2.3 AI Offloading to Accelerators

All described approaches execute the AI service on the main CPU. However, state-of-the-art AI algorithms are often offloaded to hardware accelerators, e.g., GPUs or NPUs, because of their ability to perform highly parallel computations on a large number of computing cores, yielding significantly better performance. AI offloading is not only required in data centers or high performance clusters. Sun et al. [38] analyzed Android apps which use AI algorithms and discovered that 54% of those apps utilize the GPU for acceleration purposes. Thus, practical enclave-based security architectures should enable the offloading of AI tasks to accelerators.
Partial AI offloading. Slalom [14] and YerbaBuena [35] execute Deep Neural Networks (DNN) in SGX enclaves and offload them partially to the GPU. Both approaches split the DNN computations into sensitive and non-sensitive parts. The sensitive parts remain in the SGX enclave, whereas the non-sensitive parts are offloaded. Slalom outsources the linear layers of the DNN, whereas YerbaBuena outsources the vertical layers. Both require manual effort to split the AI algorithms into sensitive and non-sensitive parts. Moreover, offloading the complete DNN achieves an acceleration by a factor of 50, whereas the partial offloading only accelerates the computations by a factor of 4 to 20 (depending on the used DNN model and the provided security guarantees).
Complete AI offloading. Offloading the entire AI task is the main goal of Graviton [30], which combines SGX with a customized GPU based on a recent NVIDIA GPU. The core idea of Graviton is to extend the Command Processor (CP) of the GPU by a crypto engine and new instructions such that the GPU memory management, which is performed by the GPU driver, can be verified by the CP. Moreover, cryptographic key material is provisioned to the CP, which can be used to establish secure channels to multiple enclaves, thus, effectively sharing the GPU between enclaves. In Graviton, enclaves can send commands, AI algorithms and sensitive data to the GPU over their secure channel, encrypted and integrity-protected. The crypto engine in the CP is only used to decrypt and verify the GPU commands, bigger data blocks are decrypted and verified by the compute engine of the GPU to reduce the performance overhead. Graviton induces a performance overhead of around 25% (on average), compared to an execution on an unprotected GPU.
Ghosh et al. [26] also aim to provide complete offloading of AI tasks from SGX enclaves to accelerators. The authors focus on DNN computations and they use Field-Programmable Gate Arrays (FPGA) as the accelerator devices. Two AES-GCM crypto engines are implemented at the accelerator, which allow for a fast encrypted and authenticated communication (over DMA and MMIO memory) between the enclave and the accelerator. In contrast to Graviton, the accelerator cannot be shared between multiple enclaves at a time. An overhead of 80% is reported for the data transfer, where the transfer time constitutes only for a small fraction (up to 6%) of the DNN inference time and thus, the overall overhead for DNN inference can be kept below 5%.
Heterogenous Isolated Executon (HIX) [16] adopts another approach to extend the protection of SGX to the GPU. The core idea of HIX is to provide a central enclave, called *GPU enclave*, which claims exclusive access to the GPU. Other enclaves, called *user enclaves*, can establish a secure channel to the GPU over the GPU enclave which allows them to shift workloads to the GPU. The GPU enclave acts as the central gatekeeper to the GPU which also allows to divide the GPU among multiple user enclaves. Changes to a commodity SGX platform must be made to support HIX: i) The GPU driver must be modified and moved from the OS to the GPU enclave. ii) New SGX instructions and structures must be added to perform access control checks. iii) The PCIe root tree must be modified such that the routing configuration can be locked, preventing an adversary from routing data to devices under his control. All data communicated between the user enclave, GPU enclave and GPU is encrypted and integrity-protected. As in Graviton, the compute engine of the GPU is utilized to encrypt and verify large data blocks. HIX achieves a performance overhead of 26.8% on average.

3 COMPARISON OF EXISTING SOLUTIONS

We compare the aforementioned solutions with respect to only the set of characteristics most relevant for AI services. Since most of the proposed solutions inherit their security and privacy features from the underlying enclave-based security architecture, we directly analyze and compare the underlying security architectures. We distinguish between architectures that are leveraged in solutions that perform the AI computations on the CPU (Subsection 3.1) and those which offload them to accelerators (Subsection 3.2), and provide a summary of our analysis in the Tables 1 and 2, respectively.

3.1 AI Computation on the CPU

	Intel SGX	AMD SEV	Sanctuary	Keystone
Main memory encrypted & integrity-protected	✓	✓	×	×
Controlled side-channel resilience	×	×	✓	✓
Cache side-channel resilience	×	×	✓	✓
Protected I/O communication	×	×	✓	✓
Performance	↓	↑	↑	↑

Table 1: Comparison of enclave-based security architectures performing AI computations on the CPU.

Main memory encrypted & integrity-protected. Storing enclave data encrypted and integrity-protected in the main memory is crucial for scenarios where AI computations are performed in public clouds where the cloud provider is not trusted. In such a scenario, physical attacks on the main memory are often considered practical. SGX and SEV provide a transparent encryption of enclave memory outside of the CPU package by leveraging the MEE and SP, respectively. Moreover, both architectures provide integrity protections, whereas SEV introduced this feature only recently with SEV-SNP [2]. While Sanctuary also provides main memory integrity protection against software attackers, Sanctuary does not defend against hardware attacks, e.g., snooping attacks on the memory bus, which are also less likely and more difficult to conduct on mobile devices, where the main memory (DRAM) is tightly integrated into the SoC. Keystone would require an additional dedicated trusted hardware unit, a memory encryption engine, to encrypt enclave code or data that leave the secure on-chip memory, while integrity protection is currently partially implemented.

Controlled side-channel resilience. Xu et al. [34] show that TEE architectures in which enclaves rely on services from the untrusted OS, e.g., for memory management or I/O services, are vulnerable to side-channel attacks in which information is leaked through data-dependent access patterns, e.g., on page faults and syscalls. SGX [3] and SEV [20] are vulnerable to information leaks when providing OS services to enclaves. Regarding AI services, Tople et al. [29] show that information leakage can be exploited to infer the classification of encrypted user data when used in an SGX enclave for inference. Sanctuary and Keystone can include drivers and OS services for handling memory management and interrupts into the enclave boundary, and thus do not rely on the untrusted OS.

In recent years, many countermeasures were proposed to hide the access patterns of a vulnerable SGX enclave from a malicious OS, e.g. [3, 21, 28, 31]. However, none of them are included in the SGX design, and all of them yield a non-negligible performance overhead. To mitigate such leaks on AI services, data-oblivious implementations of AI algorithms [22, 23, 29] have also been proposed. However, these solutions require a manual modification of the AI algorithms and also introduce a performance overhead.

Cache side-channel resilience. Side-channel attacks that exploit microarchitectural resources, such as caches, are an even more powerful threat since they allow to retrieve more fine-grained access information and do not require a compromised OS. While no end-to-end cache-based side-channel attacks mounted against AI algorithms have been shown yet, they are, in principle, also a very realistic threat for AI algorithms [15]. Cache side-channels attacks can be mitigated by either enforcing strict resource partitioning and separation between the enclaves and the untrusted system software or by applying randomization techniques. However, both are not trivial because of the hardware and performance implications. SGX and SEV do not provide any hardware-based protection against cache side-channel attacks. Sanctuary can provide protection for its enclaves by excluding the enclave memory from the shared last-level cache. Keystone deploys a cache-way based partitioning against cache side-channel attacks. However, only a coarse-grained assignment of cache ways to CPU cores is possible.

For enclave architectures without a built-in mitigation, it is up to the enclave developer to only execute side-channel resilient code in the enclave. Side-channel resilient cryptographic algorithms are e.g. provided by Intel's IPP cryptographic library within the SGX SDK. However, implementing side-channel resilient algorithms is challenging, requires verification, and cannot be generalized or automated. For AI algorithms, side-channel resilience can be provided by data-oblivious algorithms [22, 23, 29]. However, as mentioned before, they require modifying the AI algorithms and induce a performance overhead.

Protected I/O communication. In SGX, communication between an enclave and a peripheral can be compromised by a software adversary since the device drivers reside outside of the enclaves in the untrusted OS. In SEV, using virtualized I/O peripherals from the VM enclaves is possible via the hypervisor. However, Li et al. [18] show that SEV cannot guarantee the confidentiality and integrity of such I/O operations and, even worse, that I/O operations can be used to decrypt pages of VM enclaves. In principle, Keystone can be configured to provide protected I/O communication for enclaves, although this has not been shown in the paper. The use case of securely collecting user data for AI services over on-device sensors is covered in OMG [27], which is based on Sanctuary. In Sanctuary, the device drivers are included in the enclaves and used to securely connected to peripherals over the secure world.

Performance. We evaluate the performance of enclave architectures by comparing the execution of AI algorithms inside and outside of an enclave. Akram et al. [4] report a slowdown for SGX enclaves on graph workloads of 5-40% on average, whereas for SEV enclaves, only an overhead of 3-17% on average is reported. The main reason for the increased performance overhead on SGX is the limited memory size that is available to enclaves (128 MB). Thus, when large AI models are used, a high number of Enclave Page Cache (EPC) faults occur which generate overhead through cryptographic operations that are performed on the enclave pages when paging them in and out of the protected enclave memory. In SEV, the performance overhead is mainly caused by the virtualization. For Sanctuary, the runtime overhead is reported at a low 2.1% when executing a Convolutional Neural Network (CNN) algorithm inside an enclave [27]. The reported overhead does not include the time needed to setup an enclave which is neglectable if the AI algorithm runs longer than a few seconds. On Keystone, the performance overhead for Torch-based AI inference models is reported to range between -3.12% to 7.35% compared to a non-enclaved execution baseline. Keystone allows to either enforce a static maximum enclave size or dynamic resizing such that the enclave extends its virtual memory on-demand when it executes. This dynamic resizing helps reduce the initialization overhead of larger applications, thus mitigating the overall performance overhead.

Comparing the enclave architectures' performance with each other is difficult since the experiments where performed on platforms with varying computational resources and AI algorithms. However, only in SGX, the memory size available to enclaves is limited by hardware. We indicate this limitation of SGX in Table 1 by using the ↓ icon for SGX and the ↑ icon for all other architectures.

3.2 AI Offloading to Accelerators

We analyze solutions that offload AI computations to HW accelerators, namely Graviton, HIX and the work by Ghosh et al., with a focus on another set of characteristics. The challenges of secure offloading to accelerators are related to those of secure communication with sensors described in Subsection 2.2. In both cases,

	Graviton	**HIX**	**Ghosh et al.**
Location of HW changes	Accelerator (GPU)	CPU	Accelerator (FPGA)
Unmodified drivers	✓	×	✓
Accelerator sharing	✓	✓	×
Cache side-channel resilience	×	×	✓
Performance	↓	↓	↑

Table 2: Comparison of enclave-based security architectures offloading AI computations to an accelerator.

secure communication channels to the enclaves are required, yet this is even more complex with accelerators since they additionally perform Direct Memory Access (DMA) to the main memory. We emphasize that all analyzed solutions are based on SGX, and thus directly inherit the protection features and shortcomings from SGX described in Subsection 3.1.

Location of HW changes. Graviton changes the hardware of the dedicated GPU, Ghosh et al. implement the required hardware support on a FPGA accelerator device, while HIX demands changes of the SGX subsystem and the PCI root tree hardware. We argue that modifications at the accelerator end are far more reasonable, particularly for FPGAs as presented by Ghosh et al., in contrast to changes at the CPU hardware, since third parties are traditionally not able to modify or extend hardware components from Intel.

Unmodified drivers. Ideally, the solution only requires minimum changes to the accelerator drivers, which is the case for Graviton as well as for Ghosh et al. In HIX, most of the GPU driver is removed from the OS and isolated in the privileged GPU enclave, thus, the driver needs to be reimplemented as a user-space driver which produces a considerable porting effort.

Accelerator sharing. Sharing an accelerator between enclaves allows to utilize computational resources more efficiently. Graviton and HIX provide a sharing of the GPU between multiple enclaves. In HIX, all enclaves that utilize the GPU are managed by the GPU enclave, and thus, a single point of failure is created in the SGX system. Ghosh et al. always assign the accelerator to a single enclave.

Cache side-channel resilience. When accelerators are shared between enclaves, cache side-channel attacks become possible also on the accelerator side, e.g., when the accelerators perform cryptographic operations [36, 37] or in principle also on AI computations [15]. Graviton and HIX provide accelerator sharing but do not consider side-channel attacks in their threat models. The work from Ghosh et al. does not provide accelerator sharing, and thus is not vulnerable to side-channel attacks.

Performance. In contrast to performing the AI computations in a CPU enclave, outsourcing them to an accelerator (insecurely) can boost performance by a factor of 50 [14]. Securing the offloading further incurs some performance overhead, which is largely dominated by the secured data transfers between the CPU and accelerator. Graviton and HIX report an average inference overhead of 25-30% compared to an unprotected execution on the GPU, while Ghosh et al. report an overall inference overhead of less than 5%, which significantly outperforms both Graviton and HIX.

4 RESEARCH DIRECTIONS

Through our comparison of enclave-based AI protection solutions, we observe that existing architectures indeed fall short in providing the features that AI services require to meet their pertinent security and privacy specifications. We highlight the following research directions, aiming for more suitable enclave-based security architectures in the future.

Customizable and flexible enclaves. As shown, all security architectures to date adopt a *one-size-fits-all* enclave approach. They provide a specific type of enclave, where the developer is expected to adapt the sensitive AI services to the available enclaves' features and requirements, which is limiting and cumbersome. Ideally, it would be more practical if the developer can deploy unmodified applications out of the box to enclaves with minimum effort, with the option to also include unmodified complete drivers into the enclave to avoid dependencies on the untrusted OS/hypervisor. This requires an enclave-based architecture that offers different types of enclaves whose boundaries and privileges can be customized, by design, according to the requirements of the pertinent service. Furthermore, seamless support for dynamic enclave sizing at run-time would also be required to adapt to different AI workload sizes.

Configurable side-channel protection. While some of the security architectures provide some protection mechanisms against side-channel attacks (both cache-based and OS controlled channels), such attacks are not even considered in the threat model of other architectures, such as SGX or TrustZone. Thus, the provided solutions for these architectures are either impractical, heavily influence the OS or incur heavy overheads, since they are not integrated by design. Keystone provides cache-way based partitioning by design, though it remains very coarse-grained. While data-oblivious AI implementations are an alternative approach, ideally, developers do not want to modify and customize the implementations for enclave deployment, but instead expect that enclaves cater for their implementations. Moreover, such an approach is neither generalizable nor automated (i.e. is specific to every AI algorithm) and implementations would need to be verified. Thus, it is required that the architecture itself integrates the side-channel protection mechanisms that take into account both scenarios where the AI workload runs on the CPU or is offloaded to accelerators. Furthermore, it is desired that these protection mechanisms are flexibly configured per-enclave individually, e.g., enabled/disabled for enclaves independently and configured with fine granularity and exclusively to individual enclaves.

Protected I/O communication. With AI services becoming ubiquitously available on our everyday devices, security architectures must incorporate mechanisms to provide confidential and integrity-protected sensor data to the AI algorithms. They should allow to assign sensors and peripherals temporarily and exclusively to enclaves via protected communication channels, thus guaranteeing that an adversary is not able to intercept this communication, and that a software adversary is unable to misconfigure or forge sensor data to send adversarial input to the AI algorithm. Protecting against physical attacks that directly target the sensors is, however, a more complicated (likely impossible) challenge to tackle.

Efficient and secure AI workload offloading. When security architectures aim at providing performance for protected AI systems that remains on par with that of unprotected counterparts, then they will need to integrate mechanisms by design to efficiently and securely offload AI workloads to hardware accelerators, while also flexibly and exclusively binding individual enclaves to these accelerators. Unfortunately, this poses multiple challenges beyond securing I/O communication (described above) and is currently lacking in existing architectures. Firstly, hardware accelerators like GPUs communicate with CPUs in complex protocols over the internal device memory and DMA to the main memory. Secondly, the design and implementation of most modern GPUs are proprietary and closed-source. Even though reverse engineering efforts can reveal a lot about their internal workings, designing a security architecture that takes GPUs into account is difficult without knowing all the vendor-specific proprietary details. Lastly, GPUs have their own security issues and information flow leakage channels that need to be considered as well. Furthermore, mechanisms to flexibly and securely (with side-channel resilience) share different cores/resources/logic on a single accelerator device, e.g., GPU or FPGA, among different enclaves are also essential, since for practical scenarios maximum utilization of such resources will always be sought after.

5 CONCLUSION

In this paper, we introduced and analyzed enclave-based security architectures which were used to protect sensitive AI services. Through our analysis, we identified the most critical shortcomings of existing enclave architectures regarding their suitability to protect AI services. In doing so, we shed light on some of the most significant requirements and research directions for enabling future security architectures to support emerging AI services.

ACKNOWLEDGMENTS

This work has been supported by the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS) and by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297.

REFERENCES

[1] ARM Limited. 2008. Security technology: building a secure system using TrustZone technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.

[2] Advanced Micro Devices. 2020. AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. *White paper* (2020).

[3] A. Ahmad et al. 2018. OBLIVIATE: A Data Oblivious Filesystem for Intel SGX. In *Annual Network and Distributed System Security Symposium, NDSS*.

[4] A. Akram et al. 2019. Using Trusted Execution Environments on High Performance Computing Platforms. (2019).

[5] C. Göttel et al. 2018. Security, performance and energy trade-offs of hardware-assisted memory protection mechanisms. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE.

[6] Dessouky et al. 2020. Enclave Computing on RISC-V: A Brighter Future for Security?. In *1st International Workshop on Secure RISC-V Architecture Design Exploration (SECRISC-V), co-located with ISPASS-2020*.

[7] D. Kaplan et al. 2016. AMD Memory Encryption. *White paper* (2016).

[8] D. Lee et al. 2019. Keystone: A Framework for Architecting TEEs. *arXiv preprint arXiv:1907.10119* (2019).

[9] D. Steinkraus et al. 2005. Using GPUs for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition*. IEEE.

[10] E. Hesamifard et al. 2018. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies* (2018).

[11] F. Brasser et al. 2018. VoiceGuard: Secure and Private Speech Processing.. In *Interspeech*.

[12] F. Brasser et al. 2019. SANCTUARY: ARMing TrustZone with User-space Enclaves. In *NDSS*.

[13] F. McKeen et al. 2013. Innovative Instructions and Software Model for Isolated Execution. In *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM.

[14] F. Tramèr et al. 2018. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287* (2018).

[15] H. Naghibijouybari et al. 2018. Rendered Insecure: GPU Side Channel Attacks Are Practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery.

[16] I. Jang et al. 2019. Heterogeneous Isolated Execution for Commodity GPUs. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*.

[17] L. Hanzlik et al. 2018. Mlcapsule: Guarded offline deployment of machine learning as a service. *arXiv preprint arXiv:1808.00590* (2018).

[18] L. Mengyuan et al. 2019. Exploiting Unprotected I/O Operations in AMD's Secure Encrypted Virtualization. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association.

[19] M. Abadi et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*.

[20] M. Morbitzer et al. 2018. SEVered: Subverting AMD's Virtual Machine Encryption. In *Proceedings of the 11th European Workshop on Systems Security (EuroSec)*. Association for Computing Machinery.

[21] M. Shih et al. 2017. T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs. In *Network and Distributed System Security Symposium 2017 (NDSS'17)*. Internet Society.

[22] N. Hynes et al. 2018. Efficient Deep Learning on Multi-Source Private Data. *CoRR* abs/1807.06689 (2018). http://arxiv.org/abs/1807.06689

[23] O. Ohrimenko et al. 2016. Oblivious multi-party machine learning on trusted processors. In *25th USENIX Security Symposium (USENIX Security 16)*.

[24] P. Jauernig et al. 2020. Trusted Execution Environments: Properties, Applications, and Challenges. *IEEE Security & Privacy* 18, 2 (2020), 56–60.

[25] R. Kunkel et al. 2019. Tensorscone: A Secure Tensorflow Framework using Intel SGX. *arXiv preprint arXiv:1902.04413* (2019).

[26] S. Ghosh et al. 2020. A >100 Gbps Inline AES-GCM Hardware Engine and Protected DMA Transfers between SGX Enclave and FPGA Accelerator Device. Cryptology ePrint Archive, Report 2020/178. https://eprint.iacr.org/2020/178.

[27] S. P. Bayerl et al. 2020. Offline model guard: Secure and private ML on mobile devices. *DATE 2020* (2020).

[28] S. Shinde et al. 2016. Preventing Page Faults from Telling Your Secrets. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS)*. Association for Computing Machinery.

[29] S. Tople et al. 2018. Privado: Practical and Secure DNN Inference. *CoRR* abs/1810.00602 (2018). http://arxiv.org/abs/1810.00602

[30] S. Volos et al. 2018. Graviton: Trusted execution environments on GPUs. In *13th USENIX Symposium on Operating Systems Design and Implementation*.

[31] T. Hunt et al. 2016. Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association.

[32] T. Lee et al. 2019. Occlumency: Privacy-preserving Remote Deep-learning Inference Using SGX. In *The 25th Annual International Conference on Mobile Computing and Networking*.

[33] Y. Jia et al. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*.

[34] Y. Xu et al. 2015. Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems. In *2015 IEEE Symposium on Security and Privacy*.

[35] Z. Gu et al. 2018. YerbaBuena: Securing Deep Learning Inference Data via Enclave-based Ternary Model Partitioning. *arXiv preprint arXiv:1807.00969* (2018).

[36] Z.H. Jiang et al. 2016. A Complete Key Recovery Timing Attack on a GPU. In *2016 IEEE International Symposium on High Performance Computer Architecture*.

[37] Z.H. Jiang et al. 2017. A Novel Side-Channel Timing Attack on GPUs. In *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI '17)*. Association for Computing Machinery.

[38] Z. Sun et al. 2020. Mind Your Weight (s): A Large-scale Study on Insufficient Machine Learning Model Protection in Mobile Apps. *arXiv preprint arXiv:2002.07687*.

[39] RISC-V Foundation. 2019. The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, DocumentVersion 20190608-Priv-MSU-Ratified". https://riscv.org/specifications/privileged-isa/.

[40] Google. 2020. Biometrics. https://source.android.com/security/biometric.

[41] D. Kaplan. 2017. Protecting VM Register State with SEV-ES. *White paper* (2017).

[42] PWC. 2020. The Essential Eight. https://www.pwc.com/gx/en/issues/technology/essential-eight-technologies.html.

[43] statista. 2019. Leading ten areas retailers are using Artificial Intelligence (AI) in their business in the United Kingdom (UK) as of 2019. https://www.statista.com/statistics/1026052/artificial-intelligence-retailers-area-of-use-in-the-united-kingdom-uk/.