MISSIONE 4
ISTRUZIONE
RICERCA

SPOKE 1
FUTURE HPC & BIG DATA
FLAGSHIP 4:
Survey of state-of-the-art approaches and gap analysis of trustworthiness, security, privacy

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

EXECUTIVE SUMMARY

Information security is today emerging as a crucial dimension for new classes of HPC and Big Data applications, which increasingly involve massive amounts of privacy-sensitive data handled by the computing facilities.
Flagship 4 has been expressly targeted to innovative technological solutions enabling multi-tenancy HPC/Cloud platforms with strong security and data privacy guarantees. As its main activities, the flagship involves several multi-faceted research challenges and technical objectives, ranging from hardware-level security primitives and the definition of a reference architecture for a RISC-V based trusted execution environment, up to security protocols, software support and tool flows, provisioning infrastructures, multi-tenancy support in HPC/Cloud environments. as well as key software technologies like trustworthy AI and Federated Learning (FL), a promising approach for improved AI systems that do not compromise the privacy of final users and the legitimate interests of private companies.
This deliverable addresses the evaluation of state-of-the-art approaches and the gap analysis for trustworthiness, security, privacy in HPC and Big Data environments.
In particular, we have identified the key areas of interest in the state of the art, which are instrumental to the survey of existing approaches and the gap analysis due at Month 8 as part of Milestone 4. These areas include:

- RISC-V Trusted Execution Environments,
- accelerator (FPGA) oriented TEE support,
- secure virtualization,
- Federated Learning,
- trustworthy AI,
- social media data analysis,
- numerical analysis,
- trusted distributed workflows, and
- stochastics models.

As an outcome of the above review, the participants identified a precise list of technological gaps that need to be addressed to effectively reach the objectives of the Flagship.

Flagship 4 / Deliverable 4: Survey of state-of-the-art approaches and gap analysis

Elena Baralis[1], Tania Cerquitelli[1], Alessandro Cilardo[2], Iacopo Colonnelli[3], Domenico Cotroneo[2], Alessandra De Rossi[3], Luigi De Simone[2], Roberto Esposito[3], Danilo Giordano[1], Nikhil Jha[1], Marco Mellia[1], Maurizio Munafò[1], Eliana Pastor[1], Roberto Pietrantuono[2], Mirko Polato[3], Laura Sacerdote[3], Luca Vassio[1], and Cristina Zucca[3]

[1]Politecnico di Torino, { elena.baralis, tania.cerquitelli, danilo.giordano, mellia, maurizio.munafo, eliana.pastor }@polito.it
[2]Università degli Studi di Napoli Federico II, { acilardo, cotroneo, luigi.desimone, roberto.pietrantuono }@unina.it
[3]Università degli Studi di Torino, { iacopo.colonnelli, alessandra.derossi, roberto.esposito, mirko.polato, laura.sacerdote, cristina.zucca }@unito.it

April 2023

1 Introduction

Information security is today emerging as a crucial dimension for new classes of HPC and Big Data applications, which increasingly involve massive amounts of privacy-sensitive data handled by the computing facilities. Flagship 4 has been expressly targeted to innovative technological solutions enabling multi-tenancy HPC/Cloud platforms with strong security and data privacy guarantees. As its main activities, the flagship involves several multi-faceted research challenges and technical objectives, ranging from hardware-level security primitives and the definition of a reference architecture for a RISC-V based trusted execution environment, up to security protocols, software support and tool flows, provisioning infrastructures, multi-tenancy support in HPC/Cloud environments. as well as key software technologies like trustworthy AI and Federated Learning (FL), a promising approach for improved AI systems that do not compromise the privacy of final users and the legitimate interests of private companies. The objectives that have been identified for this flagship will be evaluated in several application domains, from privacy-preserving data analytics to Internet and social media data analysis. As one of the objectives that we set for this flagship

project, we will extend the FL paradigm and allow it to work in a black-box setting, where the federated model is built by ensembling local models. These techniques will be applied to real-world settings, including the analysis of large-scale dataset like those typically derived from social media measurements and networks. The participants have already identified a range of industrial players potentially benefiting from the flagship results. Some of the large companies in the CN have expressed their interest in the activities related to the flagship, involving resource optimization, federated learning, fault recovery, and on-premise and on-cloud resource management, as well as hardware/software-level security primitives for trustworthy computing, security algorithms and protocols for confidentiality and attestation.

In particular, we have identified the key areas of interest in the state of the art, which are instrumental to the survey of existing approaches and the gap analysis due at Month 8 as part of Milestone 4. The key areas for FL4 include RISC-V Trusted Execution Environments, accelerator (FPGA) oriented TEE support, secure virtualization, Federated Learning, trustworthy AI, social media data analysis, numerical analysis, trusted distributed workflows, and stochastics models.

2 RISC-V Trusted Execution Environments

As a first research line addressed by Flagship 4, we will investigate the low-level architectural building blocks enabling trustworthy computing. In the spirit of Spoke 1, the Flagship is particularly interested in exploring solutions based on the open hardware philosophy, particularly the RISC-V royalty-free processor specification. We will start by first reviewing and understanding security-related aspects related to low-end RISC-V cores. In fact, the pervasive role of the smart and connected devices is making the dependability of the entire technology stack, from the electronics up to user interfaces, crucial for a growing spectrum of scenarios, from industrial applications to automotive and transports, domotics, and even defense. In these domains, security aspects of embedded and industrial systems are potentially very critical, as a successful attack can lead to a loss of safety and possibly catastrophic consequences, such as an airplane crash, plant controller subversion [108], unreported voltage or temperature overshoots. Clearly, the trustworthiness of the underlying compute devices is of paramount importance in all of the above application scenarios.

While established design techniques for the above types of systems, from simple circuits to complex multi-core processors, effectively address aspects like power consumption, working frequency, and area occupancy, security is often ignored or seen as an add-on feature. Security measures are usually implemented in software after the physical design is closed, providing few to no hardware defense mechanisms. Such an approach can be justified by several factors, mainly the ease of access and development of software over hardware as well as the manufacturer's intellectual property rights over their proprietary architectures and implementations. On the other hand, extending conventional design methodolo-

gies with security-aware processes can lead to products that are more resilient to attacks, providing native protection mechanisms capable of effectively mitigating or even eliminating system vulnerabilities, hence reducing damages and recovery costs. This integration is naturally enabled by open hardware solutions and encouraged by the growing awareness of security issues.

Developing ad-hoc system implementations with built-in protection mechanisms for confidentiality, integrity, and isolation can both enable shielding from physical attacks (e.g., bus tampering, peripheral attacks, power analysis/side channels [104, 75]) and enable new types of software defenses based on dedicated hardware-level architectural extensions. The Flagship will address the latter opportunity, relying on the non-proprietary RISC-V Instruction Set Architecture (ISA) specification. In this context, we are particularly interested in microcontroller-class RISC-V implementations. Specifically, we aim at introducing integrated defense mechanisms serving as a baseline for establishing Trusted Execution Environments (TEEs), which are suitable for resource-constrained microcontroller-class systems. TEEs are increasingly becoming a pillar in current security architectures, ranging from server-class facilities to embedded systems, with Intel SGX and ARM TrustZone being the most prominent examples of proprietary TEE solutions. They provide an isolated execution environment based upon the paradigm of trusted computing.

2.1 RISC-V specification

In this subsection we shortly review a few relevant aspects of the RISC-V open-source instruction set architecture. The RISC-V specification, based on the Reduced Instruction Set Computer (RISC) paradigm, has been very successful recently, because it fundamentally extends the open-source concept to the hardware domain, enhancing cooperation, sharing and cost reduction, granting at the same time flexibility for user implementations. RISC-V comes with its own specification [214, 215], which defines processor registers, exception model and operation codes, without going any further into specific implementations. A number of cores have already been implemented and tested, including the *CVA6* 64-bit application processor [231], and various embedded 32-bit cores, e.g. *CV32E40P*, *Ibex* and the SiFive Essential family. A core generator, named *Rocket core*, was also introduced [111]. Furthermore, RISC-V includes several extensions. Some are mandatory (e.g. Integer extension), while others can be supported only if necessary. Additional custom extensions can be developed for specific purposes, such as flow control and bit manipulation, e.g. [51, 105].

RISC-V hardware threads (*harts*) always run at a certain privilege level used to provide protection between different components of the software stack. All implementations must provide Machine mode (M mode), as this is the only mode that has unrestricted access to the whole machine. Microcontroller-class RISC-V cores suitable for deeply embedded applications may provide only M-mode, although this choice cannot natively provide any protection against malicious application code. Therefore, implementations targeted at embedded systems typically support User mode (U Mode) as well. Application-class cores tar-

geted at complex systems also support Supervisor mode (S Mode), reserved for privileged kernel code.

RISC-V resources can be accessed by means of processor *control and status registers* (CSRs). Some of these registers can only be accessed by M-mode programs (memory protection, interrupt, hart information) while a subset of them can be made available to S mode as well.

2.2 RISC-V memory isolation mechanisms

Enhancing isolation properties in computing systems can lead to attack surface reduction, so that compromising a portion of the system does not break the entire system. Therefore, RISC-V provides native support for physical memory isolation in order to allow processes with machine privilege level (Machine mode) to properly manage memory resources. The physical memory is hence partitioned in memory *regions* allocated to specific User/Supervisor mode processes. The privileged software in charge of dynamically splitting the memory and assigning region permissions is usually referred to as a System Software.

The Physical Memory Protection (PMP) unit is the hardware component responsible for the enforcement of such isolation mechanisms. The System Software interacts with the Machine mode Control and Status registers (MCSRs) associated with the PMP unit, namely PMP configuration registers (`pmpcfgx`) and PMP address registers (`pmpaddrx`, with `x` ranging from 0 to 15). RISC-V 32-bit profile includes sixteen 8-bit configuration registers and sixteen 32-bit address registers. The `pmpcfgx-pmpaddrx` pair distinctively defines a PMP *entry*, or *rule*, which basically specifies an isolated portion of physical memory called region. The address register contains an encoded address space for such a region, which is decoded by the PMP unit itself every time an access/write/read operation is performed in that address space by any running hart. Field `Address` (`A`) in the `pmpcfgx` register indicates the specific encoding (None, NA4, NAPOT, TOR) for the `pmpaddrx` register. If None encoding is selected, the PMP entry is considered disabled or empty. Bits `W`, `R`, `X` in the `pmpcfgx` register respectively stand for write, read, and execution permissions. Any write/read operation on that region must match the right permissions in order not to raise an access fault exception. The PMP unit is capable of isolating Supervisor and User mode processes from Machine mode processes (e.g. System Software). In higher-end systems, we expect the System Software to be the only one running in Machine mode. However, there are computing systems (e.g. microcontroller-class systems-on-chip) where multiple Machine mode processes could coexist, making it highly desirable to also enforce isolation among Machine mode defined regions. In order to enforce PMP rules to M-mode processes, RISC-V includes the `Locked` (`L`) bit in the `pmpcfgx` register. If this bit is asserted, the PMP entry is valid for the M-mode code as well. For security reasons, this also implies that the `pmpcfgx-pmpaddrx` register pair will not be modified until the next hard reset.

PMP also provides a priority system: a PMP entry always overwrites an overlapping `pmpcfgx-pmpaddrx` pair having a larger index. Some Trusted Exe-

cution frameworks, like Keystone, allocate the last entry for the main operating system and the first one for the System Software (Secure Monitor).

LowRISC group contributed to the native RISC-V isolation mechanism by introducing *bijective* isolation among differently privileged processes. They enhanced the PMP unit [106] (ePMP) in their deeply embedded core [122] by means of a new MCSR called `mseccfg`. The register only defines three new bits used to change the PMP behaviour:

- The Rule Locking Bypass (RLB) bit, if asserted, allows the Locked PMP entries to be deleted or edited.
- The Machine Mode Whitelist Policy (MMWP) bit is a sticky bit, meaning that once asserted it cannot be reset anymore. If this is the case, an M-mode code will not be able to access memory regions which are not covered by any PMP entry.
- The Machine Mode Lockdown (MML) bit, if asserted, makes it possible to define shared memory regions, S/U-mode only-access regions, and M-mode only access regions.

2.3 TEE technologies

Trusted Computing refers to computer systems for which an entity –either a human user, or a local or remote program– has some level of assurance that the computer system is behaving as expected [135]. It typically relies on the definition of an appropriate trusted computing base (TCB), i.e. the set of protection mechanisms within the computer system, including hardware, firmware, and/or software one must rely upon for the assurance guarantees to hold [77, 158]. Trusted computing is relevant for a plethora of application fields, from biometrics [39] to peer-to-peer networks [179]. Trusted computing mechanisms directly support the establishment of *trusted execution environments* (TEEs). A TEE is an inherently trusted environment containing a small running kernel which provides a reduced interface to the main untrusted operating system (denoted Rich Execution Environment, REE) and to other untrusted processes. Having a small running system in the TEE reduces the attack surface, making the TEE suitable for running security-critical applications, providing secure I/O, and enforcing isolation, integrity, and confidentiality for both code and data.

Memory isolation mechanisms can be exploited as a support for implementing TEEs, possibly complemented by security measures at the electronic design level [68, 196, 126, 169], which are orthogonal to TEEs. Below we give a short overview of several solutions and implementations applied on various architectures which are relevant for the scope of this deliverable.

A prominent example of TEEs is Intel Software Guard Extensions (SGX), an Intel ISA extension aiming to provide security guarantees in terms of confidentiality and integrity in untrusted environments for high-end processors [45]. Intel implements its TCB almost entirely in hardware (Page Walkers, Fault Handlers, TLBs) and microcode, partially delegating some features to trusted

software and partitioning the memory in a non-trusted memory and a trusted memory (Processor Reserved Memory) used to host secure applications. Intel introduced the popular concept of *enclaves*, which is basically a safe container in which a high-level security-critical software is loaded along with its data. Every enclave grants confidentiality and integrity to its application, defending it from external environments, including a potentially untrusted Operating System.

Another popular TEE in IoT and embedded systems is ARM TrustZone [164, 163], a collection of hardware modules that can be used to partition system resources between a *secure world*, which hosts a secure container, and a *normal world*, which runs an untrusted software stack [46]. An ARM processor core can switch between the normal world and the secure world when executing code. Furthermore, it has to implement an additional memory protection unit called Security Attribution Unit, which takes a CPU address and establishes if that address is safe or not, similarly to the RISC-V PMP. The ARM TrustZone high-end solution provides a system software module called Secure Monitor responsible for mediating between secure and non-secure worlds. On the other hand, TrustZone-compliant microcontroller-class systems provide ad-hoc instructions to switch from a world to another and allow communication between secure and non-secure world, reducing the software overhead incurred by the Secure Monitor.

Open-source architectures, like RISC-V, are also being extended with TEE support. MultiZone [184] is a TEE targeted at RISC-V relying on a generalization of the ARM TrustZone philosophy. Instead of having two worlds, MultiZone introduces multiple worlds, or *zones*, which can isolate specific portion of running code and data, like libraries in a similar way to SGX secure containers. A strong point in MultiZone is flexibility, as its TCB consists of a nanokernel and some communication and scheduling modules, implying decoupling from hardware solution, making MultiZone extremely portable. While MultiZone is a commercial solution, noncommercial TEEs are under development for RISC-V. Keystone [110] is an open-source TEE framework for RISC-V, supporting 32-bit and 64-bit architectures and requiring all three privilege modes (M, S, U) in order to support dynamic isolation. Keystone is inspired by SGX and Sanctum [47], another high-end processor TEE framework designed for RISC-V before the introduction of PMP.

Keystone inherits the concept of enclave from SGX and implements it in a simple and effective way by means of the PMP native mechanisms, but instead of relying on microcode it provides a Secure Monitor, inspired both by Sanctum and TrustZone. This means that the Keystone TCB is completely software, therefore no physical and virtual memory protection is integrated on chip. Moreover, the design choices made by Keystone allow the Secure Monitor to be written in C and easily verified, as long as its footprint stays limited. It is the only Machine mode software running and it is responsible for PMP configuration, and hence for process isolation and enclave lifecycle management.

Identified gaps. As highlighted above, our aim in this particular branch of the Flagship activity is to explore TEE support, most notably effective memory isolation mechanisms, for resource-constrained RISC-V implementations. Based on the review we conducted, despite enabling isolation for User mode processes, the improvements brought by the ePMP concept are not enough to support trusted execution in microcontroller-based systems which do not support a Supervisor Mode. Since we are interested in complex systems, possibly running multiple privileged software components (e.g. drivers, embedded Operating Systems, Real-Time Operating Systems [229, 123, 14, 81]), which would all run in Machine mode, we see here a technological gap to be filled. In fact, simply locking entries in the PMP ensures strict isolation, but it reduces the System Software's capability of creating and deleting new PMP entries, preventing the dynamic management of the trusted environment. In the project we will address these gaps by introducing new mechanisms allowing a more versatile Machine mode PMP region management, while keeping the attack surface restrained.

3 Accelerator oriented TEE support

Accelerator-based and special-purpose machines, e.g. based on Graphics Processing Unit (GPU) and Field-Programmable Gate Array (FPGA) technologies, can play a key role in sustaining the evolution of high-performance computing technologies, compared to standard platforms like general-purpose processors. In fact, accelerators have proved to provide substantial speedups in HPC application domains ranging from bioinformatics to financial computing, from storage to artificial intelligence.

In terms of security and privacy, this trend poses however a crucial challenge, as user data inherently need to leave the main processor units within the system and move to peripheral devices, which act as a black box in security terms as seen from the system integrator's and user's perspective. This is particularly true of distributed, cloud-oriented infrastructures. That poses a crucial hurdle, since exposure of sensitive data raises privacy concerns which may discourage users of accelerated HPC platforms because of their opaque setting.

A mostly theoretical solution is provided by pure cryptography-based approaches, like homomorphic encryption [76] and garbled circuits [224]. They completely remove any hardware component from the trust compute base (TCB), but unfortunately they only fit special types of operations and have prohibitive compute/communication requirements for many practical tasks [40, 94]. In particular, their use for complex workloads such as Machine Learning algorithms is out of question in the current state of the art.

On the other hand, Trusted Execution Environments (TEEs) may play a role in addressing the above limitations as a pragmatic solution, still requiring a third party in the system TCB, but limiting the trust assumption to the processor manufacturer. In particular, the commercial solution for privacy-preserving computing introduced by Intel, based on the notion of secure SGX enclaves, only includes Intel processors in the TCB and allows critical data and compu-

tation to be securely offloaded to cloud services in untrusted settings, as long as Intel processors are considered trusted. AMD Secure Encrypted Virtualization (SEV) provides another solution from the other major general-purpose processor manufacturer. However, CPU-based solutions suffer from inherent limitations, especially when targeting high-performance applications. For example, Intel SGX enclaves are essentially not meant for high-performance computing, with the basic form of enclave offering only 256MB of memory in the early versions of SGX. Most importantly, with processor-based TEE dedicated accelerators, like GPU or FPGA devices, are completely excluded by the secure perimeter of CPU-side enclaves. Just a few academic works have recently addressed these limitations [209, 230]. In a sense, the crucial requirement for user's data privacy seems to inherently clashes with the need for special-purpose acceleration and dedicated machines in HPC environments.

Identified gaps. We see two main gaps to be addressed in the area of trustworthy HPC:

- CPU-based TEE solutions are not a good fit for the requirements of HPC/big data applications;
- no commercial solutions exist for accelerator-based trustworthy HPC, e.g. based on FPGA technologies.

4 Secure virtualization

The role of virtualization technology is widening today, from cloud computing systems to critical industrial systems in several domains (e.g., railways, avionic, automotive) [44], due to its ability to reduce *SWaP-C* factors (size, weight, power, and cost) by consolidating multiple software stacks on the same system-on-a-chip (SoC)[44, 1].

In cloud computing, virtualization allows distinct customers to hire online computing resources for their own purposes. By doing so, specific applications (Software as a Service—SaaS), development platforms (Platform as a Service—PaaS), or complete virtual machines with networking components and storage capabilities (Infrastructure as a Service—IaaS) can be requested from the cloud operator on a pay-per-use basis. In this scenario, the allocation of the hardware resources changes at run-time, taking into consideration several factors such as the user demand (i.e., the utilization) and the plan paid by the user.

Similarly, in the industrial domain, isolation properties of virtualization are appealing for functional safety standards (e.g., DO178C for avionic [176], ISO 26262 for automotive [91], etc.), which recommend providing evidence on temporal, memory, and fault isolation among applications, sharing the same computing infrastructure. In this context, predictability is preferred over utilization, thereby the resources are often allocated statically.

Virtualization is also considered one of the most promising technologies to streamline the adoption of MPSoCs (Multi-Processor Systems on Chip) [43, 15, 220, 99, 137]. These SoCs are characterized by heterogeneous hardware components such as APUs for general-purpose computation, FPGAs (Field Programmable Gate Array) for high-performance computing (HPC), RPUs for real-time and safety computation, and GPUs (Graphic Processing Units) for parallel and graphical computations. The heterogeneity of these SoCs guarantees high performance, scalability, and reconfigurability, but their inherent complexity makes them not easy to use. In order to improve the predictability and usability of heterogeneous hardware, virtualization is therefore seen as a key technology.

More recently, a growing interest sees the adoption of cloud technologies in Industry 4.0, offering to host critical applications as a service. EU initiatives and projects such as Digitale Schiene Deutschland [148, 199] and SECREDAS [70] have looked into cloud computing for hosting safety-relevant railway applications. TransVital [198] and DS3 [190] are up-comping platforms from Thales and Siemens to support safety-critical railway applications, such as interlocking and radio block centre, in a SIL4 Cloud [82].

The widening adoption of virtualization makes its security extremely important, especially in an industrial domain where the violation of isolation properties can lead to catastrophic consequences [112].

4.1 Hardware virtualization

Hardware virtualization provides an abstraction of physical hardware resources in multiple virtualized hardware resources to increase utilization and reduce running costs. Today, hardware virtualization is linked to the possibility of running multiple Operating Systems (OS) on the same hardware platform, which means virtualizing all the hardware resources (i.e., Virtual Machine (VM)) necessary for an OS to run. Although, hardware virtualization also includes virtualizing access to existing devices (e.g., GPU or FPGA) and emulating new devices. More specifically, the first case is generally adopted to multiplex the access to the same physical resource, such as network or disk storage. In contrast, the second case opens the possibility of interfacing with peripherals that are not available in the hardware platform. It is worth noting that the emulation in HPC platforms can be faster than the operational frequency of real physical devices opening new opportunities in cloud computing and software testing [37].

In the software stack of hardware virtualization, the hypervisor is the privileged software component that setups, manages, and abstracts the hardware resources. For that purpose, the hypervisor uses hardware extensions available in modern CPUs [140, 6] dedicated to virtualization, which introduces a novel (and higher) level of privilege for the hypervisor, new features to support hardware virtualization (e.g., a new level of translation in the page table), and a configurable structure to setup which guest conditions and instructions are sensitive for the hypervisor and thereby might call hypervisor intervention (i.e., trap and emulate).

The adoption of such virtualization technology is known in the literature as *hardware-assisted virtualization*, which takes advantage of CPU virtualization technologies (e.g., Intel VT-x [141], AMD-V [10], ARM VHE [50]) to implement *full virtualization*, i.e., emulating a complete machine to run unmodified guests.

The following sections are structured as follows.

- Section 4.2 introduces virtualization technology, taking ARM VHE and Intel VT-X as example.
- Section 4.3 shows an overview of real-time embedded virtualization challenges and state-of-the-art approaches.
- Section 4.4 shows the current threat model and attack vectors identified over the hypervisor.
- Section 4.5 provides an overview about hypervisor detection.
- Section 4.6 provides the current state of the art regarding the discovery of hypervisor vulnerabilities.
- Section 4.7 provides a gap analysis given the above discussion.

4.2 Intro to virtualization technology

In order to run VMs with unmodified guest OSes, several CPU vendors (e.g., Intel, AMD, ARM) develop hardware virtualization extensions to reduce overhead and improve the performance of virtualization.

More specifically, ARM introduces, from ARMv8, a new privileged execution mode called *EL2* (Exception Level 2) in addition to the kernel (*EL1*) and user (*EL0*) modes. To guarantee the isolation of VMs, the hypervisor runs in EL2 mode, having complete control of the hardware, while the VMs software (guest OS and user applications) runs in EL1 and EL0 modes. To limit virtualization overhead, VMs run without hypervisor intervention as far as possible, until some particular conditions occur. In the latter case, a hardware trap is activated and the hypervisor starts taking control of the hardware and eventually return to the VM after emulating the behavior expected by the VM. By doing so, the software running in EL1 works exactly as it would run without virtualization extensions. It is worth noting that if virtualization features are disabled in EL2, then processes running in EL1 have direct control of the hardware, regardless of any virtualization support, thus being transparent to non-virtualized executions. To enhance the security features of their processors, ARM developed TrustZone technology [159, 160]. The main idea is to improve the isolation capabilities by enabling two worlds, the *secure world*, where a small trusted OS runs in isolation, and a *normal world* where everything else runs. These worlds allow operating on dedicated memory regions with different privileges, and a *secure monitor* is responsible for switching between secure and non-secure execution.

Intel VT-x [90] is the target technology of several works over security, as they continue to represent the largest CPU family used in the server segment

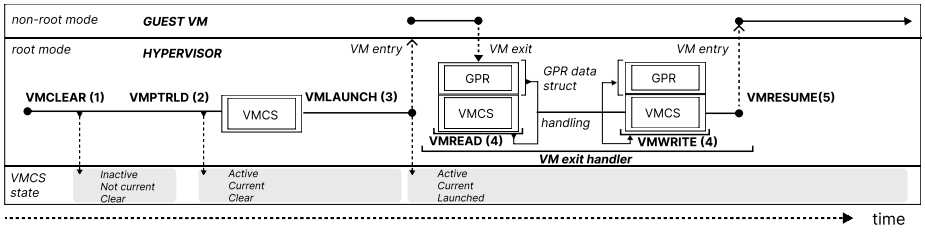

Figure 1: Workflow of a virtual machine in VTX

worldwide. For this reason, we choose to provide an overview of the hardware extensions and of the VM life cycle in Intel technology to make easier the comprehension of the threat model.

Intel VT-x and VM Life-cycle Once the virtualization is enabled (`VMXON` instruction), two operating modes are active. The hypervisor (*VMM*) operates in *root mode*, while the guest *VM*s run in *non-root mode*. The latter modes are orthogonal to traditional execution modes (long, protected, and real modes) and to privilege levels (i.e., rings). Running a new VM in *non-root mode* requires allocating and initializing in memory a particular control structure, called *Virtual Machine Control Structure (VMCS)*, linked to a specific vCPU. The VMCS, except for its first eight bytes, must be read and written by executing dedicated *VMX instructions* called `VMREAD` and `VMWRITE`, otherwise unpredictable failure modes can occur (see Section 24.11.1 in [90]). The VMCS consists of the following areas: *guest-state*, *host-state*, *control fields*, and *VM exit information*. The first two are the most important in the context of our framework and include, respectively, the processor state when the VM is suspended and resumed. Specifically, they include special-purpose registers (e.g., control registers, instruction pointers, etc.).

Fig. 1 depicts the VM lifecycle. The VMCS is initialized (`VMCLEAR` instruction, step 1 in Fig. 1) during the VM startup and subsequently loaded (`VMPTRLD` instruction, step 2 in Fig. 1). When the VMCS is loaded, its internal hardware state becomes *Active Current Clear*. In this state, the hypervisor can set up the VM, for example, by defining the events and instructions in *non-root mode* that will cause a switch to the *root mode* (i.e., a *VM exit*). Once the setup is completed, the hypervisor can launch the VM (`VMLAUNCH` instruction, step 3 in Fig. 1). Once this instruction is complete, the VMCS state becomes *Active Current Launched* and the Guest VM can run, after switching to non-root mode (called *VM entry*).

During the execution of the VM, the control can pass to the hypervisor every time a *VM exit* occurs, requiring a context switch from non-root to root mode. VM exits can occur for different reasons. Currently, Intel *x86* architecture support 69 *VM exit reasons* (Appendix C, Table 1-c [90]). Most of them are due to the execution of sensitive instructions by the VM, such as `RDMSR`, `WRMSR`, or `CRx ACCESS`. Others include VM events or conditions to be handled by the

hypervisor, such as triple fault, interrupts, and I/O port access. Finally, the hypervisor can decide to trap some VM conditions to follow the VM evolution (e.g., VM introspection [84]) or to take scheduling and resource-sharing decisions (e.g., memory deduplication [127]).

The VM exit is a key operation since it can be exploited to compromise the isolation properties of the hypervisor. Hence, we use it as the mean to submit a seed to the hypervisor and to test its operation. Let us analyze in detail the steps occurring from the VM exit up to the successive VM entry (VM resume), including the execution of the *VM exit handler* in the hypervisor (steps 4 and 5 in Fig. 1). The VM exit requires a hardware context switch from non-root to root mode, that entails: (i) to save the physical processor state in the guest-state area of the VMCS (except for general purpose registers (GPRs), saved in the hypervisor data structure), (ii) to load the new root mode processor state from the host-state area of the VMCS, including also the instruction pointer register (RIP), containing the start address of the VM exit handler. After the context switch, the VM exit handler identifies, from the VMCS, the cause of the exit and appropriately resolves it. More importantly, during the execution, the VM exit handler can access the entire VMCS (`VMREAD`, step 4 in Fig. 1), hence its control flow depends on VMCS fields. Additionally, the VM exit handler can change the VM state in the VMCS (guest-state area) (`VMWRITE`, step 4 in Fig. 1). Once the VM is resumed (`VMRESUME` instruction, step 5 in Fig. 1), the new VM state becomes operational on the physical CPU. The VMRESUME performs a new (inverse) hardware context switch, where the processor state is loaded from the guest-state area of the VMCS.

Basic Hypervisor Virtualization components. In hardware-assisted virtualization, the processor supports the virtualization at hardware level by dedicated hardware extensions (e.g., EPT, VT-d, SR-IO...). For instance, the Extended Page Table (EPT) allows the hypervisor to map the physical memory of the single VMs to the real physical memory without calling its intervention. We call Virtualization via Hardware (VH) the virtualization implemented via hardware extensions configured to not call hypervisor intervention (i.e., no software intervention). Belong to VH also the pass-through of devices (i.e., when the peripherals are allocated to the single VMs). However, even the support of hardware to virtualization, the software emulation (trap and emulate) is still present and supported in hardware-assisted virtualization for several reasons (Virtualization via Software (VS)). The current version of the architecture manual specifies 69 (Appendix C, table 1-c [90]) different codes for "basic VM exit reasons". Table 1 groups the exit reasons by the high-level context in which they occur [1]. Most of them are due to the lack of hardware support. For instance, VMX has limited support for nested virtualization, thereby the hypervisor has to emulate VMX instructions to enable nested virtualization. Again, the APIC does not allow the hypervisor to limit what CPUs a VM can interrupt via inter-processor interrupts (IPI), thereby the hypervisor needs to emulate the APIC accesses. Other CPU instructions can be intercepted and

[1] The exits reasons can belong to a more high-level context

emulated (e.g. CRx, GDTR, LDTR and MSR). Memory virtualization is virtualized via software in a few cases like Populate-On-Demand [218] where the hypervisor allows a VM to have a physical memory bigger than the real physical memory. Device virtualization and emulation are the other important reason to intercept the VM execution. Indeed, as also we mentioned in the previous sections, device emulation has surprising performance capabilities and has the power to emulate also not available devices. The interactions of devices are trapped by the hypervisor via memory (MMIO) and via IO ports (PIO).

However, the hypervisor intercepts the VM not only for emulation and virtualization. The hypervisor can decide to trap some VM conditions to follow the VM evolution (e.g., VM introspection [84]) or to take scheduling and resource-sharing decisions (e.g., memory deduplication [127]).

Table 1: Exit reasons in Intel VT-x

Context	**Exit reasons**
CPU virtualization	Exception or NMI (0, 8, 45); INIT signal (3); SIPI (4); SMI(5,6); CPU Instructions (9-17,28-29, 31-32, 36, 39, 40, 46, 47, 51, 54, 55, 57, 58-68); APIC(43, 44, 56)
I/O virtualization/emulation	External interrupt (1); IO Instruction (30); Interrupt Window (7); EPT (48, 49)
Memory virtualization	EPT (48, 49)
VT-x Exceptions	Triple fault (2); VM-entry failure (33, 34, 41); Monitor Trap Flag (37); Preemption timer (52)
Nested virtualization	VTX instructions (19-27, 50, 53)
Hypercall	VMCALL (18); VMFUNC(59)

4.3 Hypervisor for resource-constrained systems

Recently, there has been a great deal of commitment to developing new hypervisors for embedded systems and for real-time and isolation purposes [129, 115, 128, 124, 166, 3].

While the objective is almost identical in most cases, the approaches used are quite distinct. Some of these solutions are built on micro-kernel or separation kernel architectures with the aim of reducing the complexity of the hypervisor and simplifying the certification process. These are specially intended for

IoT/embedded context and, among these, we found seL4 [59, 103], NOVA [195], and PikeOS [103].

Other lightweight approaches are those based on partitioning techniques, such as Jailhouse [166], Bao [128], and Xtratum [129]. These tiny hypervisors are designed to statically partition the hardware resource to the guest VMs minimizing the hardware interference bearing the cost of less efficient use of resources.

A completely different but equally interesting approach is to modify an already existing and widely used general-purpose hypervisor to guarantee real-time requirements. This strategy enables the reuse of knowledge and the exploitation of strong communities to accelerate the use of virtualization in embedded systems. KVM and Xen are two examples [3].

Another solution adopted in the literature is to rely on hardware security mechanisms like *ARM Trustzone* [159], which guarantee strict isolation between two environments, the secure and non-secure worlds. LTZvisor and RTZVisor are ARM TrustZone-based solutions that leverage the hardware-based mechanism to implement *dual-guest* OS and also *multi-guest* OS virtualization, respectively.

4.3.1 MPSoC Hypervisors

There are some recent works that have been proposing techniques to virtualize heterogeneous platforms such as MPSoCs, featuring a programmable logic (FPGA) as well as heterogeneous processors, to realize reliable mixed-criticality systems where the isolation is guaranteed for each VM.

CHIPS-AHOy is a predictable holistic hypervisor [138] that aims to satisfy temporal predictability and high-performance requirements of software running over MPSoCs while simultaneously handling energy efficiency, thermal bound, and system lifetime. The authors' goal is to address the most relevant source of unpredictability in MPSoCs such as the memory hierarchy, the I/O subsystem, and the hardware variability by using techniques such as cache coloring, and I/O throttling. Therefore, they try to leverage platform-specific hardware such as PMUs and physical sensors to predict the effect of actuation actions in the system in order to improve the predictability of the following actions: real-time scheduler, cache coloring, I/O throttling, and reliability management.

Biondi *et al.* present the SPHERE project [20], an integrated framework to abstract the hardware complexity of MPSoCs and simplify the management of heterogeneous hardware. The idea is to extend the functionality of a hypervisor (SPHERE supports CLARE and Jailhouse hypervisors) for next-generation cyber-physical systems with real-time guarantees. The authors focus on a multi-soc scenario where the hypervisor is able to manage time-sensitive networks in presence of traffic flow with different temporal constraints. An interesting part of the work explores the possibility of using the dynamic function exchange capabilities of the FPGA (also known as dynamic partial reconfiguration) to provide efficient implementations for cryptography modules, as well as hardware acceleration for deep neural networks.

4.3.2 Deterministic I/O

I/O can be a strong source of non-determinism, especially in virtualized MP-SoCs which are characterized by many peripherals usually used simultaneously. This issue can also impact the security of virtualized systems. For this reason, companies provide hardware support for I/O virtualization: i.e. Intel VT-d [4] allows assigning I/O devices to VMs while providing VM routing for device interrupts, and SR-IOV improves the management of PCIe devices. Furthermore, there is a strong commitment in literature to find a solution to I/O virtualization in real-time environment [28, 2, 178, 102, 96].

Despite all this effort, the heterogeneity of new approaches and solutions, and the plethora of papers about isolation in virtual environments, accelerator support is still immature, as shown in [43].

4.3.3 Accelerators Virtualization

There are many techniques to virtualize the GPU and the FPGA in server environment, such as fixed pass-through, device emulation, or hardware support virtualization, but in all these cases the solutions do not focus on isolation and minimal impact, but rather on performance. Therefore, porting it to an embedded system with real-time requirements requires effort.

Virtualizing GPUs is a relatively new area of study, and although there are several proposed solutions [87, 156], it remains an open challenge. This is not only due to the heterogeneity of the GPU's architectures but also because GPU drivers are not open for modification due to intellectual property protection. All these motivations make conventional virtualization techniques not directly applicable for GPU virtualization.

Virtualization of FPGAs is also a very complex area of research. There are several surveys that attempt to describe the current state of the art in FPGA virtualization [205, 217, 21]. It is possible to distinguish two directions to FPGA virtualization in the literature: the FPGA used as a shared hardware accelerator resource between VMs [211, 219], and the FPGA used to improve the features of the hypervisor [95, 92]. Following the latter approach, several papers focus on increasing the predictability of the hypervisor leveraging FPGA for real-time purposes. For example, in [171] the authors propose a hardware component to allow interconnecting hardware accelerators to the same bus while ensuring isolation and predictability. In [175] and [174] FPGA is used to control the memory hierarchy reducing memory access latency and increasing predictability and isolation of the software. FPGA is also used to profile the memory demand of CPUs and accelerators in order to predict the temporal behavior of deployed workload[193].

4.4 Threat model and attack vectors to hypervisor

We consider the following threat models:

- a malicious VM with the aim of compromising confidentiality, integrity, and availability of the hypervisor or other VMs.
- a malicious VM with the aim of identifying the presence and version of the behind virtualization
- a malicious hypervisor (i.e., Virtual-machine base rootkits (VMBR)) that would transparently trace the actions of its VM victims

4.5 Hypervisor detection and fingerprinting

Virtualization detection techniques can reveal the presence of virtualized environments, which instead should remain transparent. Generally, these techniques infer the presence of virtualization through the timing deviations of specific machine instructions or by looking for artifacts/fingerprints left either by the VMM or the VM itself.

The interest in virtualization detection spiked up when many VM-based malware rootkits were introduced, as the detection of the VMBR (Virtual-machine base rootkits (VMBR)) might save the VM from further spying of data [23]. Moreover, the detection of virtualized environments is often the first step carried out by many attackers to narrow down and fine-tune their attack strategies on cloud infrastructure [71].

Logic of machine instructions The famous Red Pill test for identifying a VM on x86 architectures was introduced by Rutkowska, which relied on using the SIDT machine instruction to identify the differences in the Interrupt Descriptor Table (IDT) addresses in the guest and host machines. The CPUID, a machine instruction on x86 and x86_64 CPUs, can reveal the presence of VMMs [23]. Other approaches have been devised to detect virtualization through VMM fingerprints, as the system's BIOS data [48] or virtualized drivers [120] can reveal the presence of different hypervisors.

Timing attacks for virtualization detection Timing attacks for virtualization detection usually rely on measuring the overhead involved in the VM Exit operation [23], which is present only in the case of virtualized environments. The approach involves measuring the time taken for specific instruction executions or specific operations inside a VM and comparing it against the values obtained on a host machine. The `CPUID` machine instruction [90] is usually used for performing such a timing analysis, as this instruction is always known to result in a VM Exit. This instruction can be executed from the user space itself and is thus a good candidate for performing the check from a normal guest account on the machine.

Other timing-based approaches use the CPU's Return Stack Buffer (RSB), Timing analysis on the Translation Lookaside Buffer (TLB) [23], or the usage of memory virtualization features by the processor [71].

4.6 Hypervisor Vulnerabilities

The hypervisor runs with an elevated privilege on the host machines. Owing to its importance, the hypervisor becomes a preferred target among the attackers of the cloud infrastructures. Vulnerabilities in this layer can lead to an attacker running as a malicious guest user, to break into the higher privileges of the hypervisor or cause a denial of service from the host [144, 143, 142], which might be running multiple VMs owned by different cloud users.

Verifiability of the hypervisors becomes a challenge with the large and growing codebase of the hypervisors.

Below, we split the approaches of vulnerability discovery into two categories, as they target two distinct interfaces asking a different knowledge for testing.

CPU Virtualization testing. *Amit et al.* [11] propose to apply the testing environment of CPU vendors to hypervisors, however, they need an intimate awareness of x86 architecture to generate comprehensive test cases. *PokeEMU* [221] generates CPU test cases for virtual CPU implementations applying symbolic execution exclusively to an executable specification, without considering the implementation. However, its main targets are hypervisor with no hardware-assisted virtualization. Similarly, *MultiNyx* [69] generates test cases focusing on hardware-assisted virtualization, by applying dynamic symbolic execution. However, *MultiNyx* records multiple traces between VM and VMM context incurring a high performance overhead. *HyperFuzzer* [72] is a hybrid fuzzer for virtual CPUs. Both *HyperFuzzer* and *MultiNyx* are snapshot-based fuzzer. Its main difference from [69, 221] is that it avoids the overhead of a full hypervisor execution track, relying on instrumentation. Instead, it only records the program's control flow by using commodity hardware tracing. These studies construct the initial fuzzing seeds manually based on expert knowledge. In addition, they do not focus on I/O device virtualization behaviors.

Device Virtualization testing. The following studies do not mutate the VM's architectural state. This can limit their testing coverage, as the hypervisor depends on the VM's architectural state when emulating an operation. *Schumilio et al.* [182] first discover the available hypervisor interfaces via a custom OS, then they test such interfaces through a black-box fuzzer based on a custom bytecode interpreter which accelerates the input generation phase. Once again, the fuzzing seeds are built manually. *Nyx* [183] tests the hypervisor target via nested virtualization using KVM. In addition, *Nyx* uses grammar rules to specify the structure of the target emulated devices. Relying on manual input grammars per device requires manual work to specify grammar rules [139], thereby several studies record the interactions between the guest operating system and the device [139, 85, 149, 26]. *Henderson et al.* [85] selectively instrument the code of a given virtual device, and perform a record and replay of the only memory-mapped I/O (MMIO) activity of the virtual device in QEMU. *VShuttle*, *Morphuzz*, and *MundoFuzz* [149, 26, 139] fuzz the entire emulated device input interface including DMA interactions. Contrary to MMIO and PIO interactions that call the hypervisor intervention interrupting the VM (VM exit), the DMA does not interrupt the VM. Indeed, both the work [149, 26] instrument

the DMA API of the Hypervisor to target the dynamic memory regions where the DMA is working. Instead, *MundoFuzz* [139] collects IO instructions and DMA operations within the guest operating system without hypervisor instrumentation. MundoFuzz [139] fuzz the hypervisor with grammar-awareness using automatic grammar inference. Hypervisor grammars have hidden input semantics, and *MundoFuzz* finds the causal relationships between the inputs through experiments (statistical learning). Additionally, the recorded inputs could be interleaved from asynchronous events (e.g., the timer interrupts) that generate coverage noises. *MundoFuzz* deletes this noise through differential learning.

Record and replay. In fuzzing, the record and replay is an effective way to learn the grammar of the target system [139, 85, 149, 189]. However, record and replay are also adopted in security to analyze and debug execution traces. Record and deterministic Replay (RnR) is a popular architectural technique [24, 56, 41, 57, 186, 212]. The RnR injects the recorded events at the correct times, enforcing a deterministic execution (Replay). RnR is used for several reasons. For instance, when the system adopts no precise events to detect possible exploits and violations, the replay is used to verify if those events are false positives [186]. It is also used to analyze time-of-check to time-of-use race conditions [57] or to determine if systems were previously exploited once zero-day attacks are discovered [97]. RnR can be done at different abstraction layers, however, to the best of our knowledge we are the first to record and replay the VMM history in hardware-assisted virtualization solutions.

4.7 Identified gaps

- **RPU Virtualization** Both GPU and FPGA virtualization techniques are not suitable for virtualizing the RPU (Real-Time Processing Unit) since both are very hardware specific. Furthermore, GPU virtualization techniques are used to accelerate calculations in order to achieve higher performance while FPGA virtualization is used both to achieve higher performance and to expand the hypervisor features. On the other hand, the RPU is used to increase the determinism of the task that runs on it. It is therefore essential to develop virtualization techniques that focus on the requirements of predictability, security and isolation, taking into account the strengths and limitations of the RPU architecture.

- **Hypervisor misbehavior detection**. The hypervisor, as shown in Fig.1, takes some actions during a VM exit before returning the control flow to the VM. These actions include the update (i.e., VMWRITEs operations over the VMCS in Fig.1) of the guest physical CPU state (e.g., the program counter, control registers), as the emulation traps the sensitive guest instruction before its execution and returns the guest control flow as the instruction was executed (i.e., emulated).

  To be transparent, these actions should faithfully emulate a CPU physical behavior, as any misbehavior could be a way to detect virtualization or, worse, it could be an alarm of a possible exploitable bug. However,

there are no proposed solutions to test the correctness of these actions in hardware-assisted virtualization, as the current solutions are more focused on testing emulators like QEMU.

- **Discovery of CPU virtualization bugs**. The test case generation in CPU virtualization asks for a deep knowledge of the underlying hardware, as the test case includes the setting of both the mode of the physical CPU and the guest instruction or condition that will be the reason for a VM exit. Today, the most efficient approach to running a test case seems to be the reverting of a VM snapshot appositely crafted ad-hoc, which support testing and fuzzing techniques. This way, both the state of the guest physical CPU and the next instruction to run can be set. However, (1) the programming of a snapshot is not easy, as the next instruction to run depends on the specific guest CPU state (e.g., the program counter, the page table), so it asks a golden starting seed to be effective. Second, (2) the snapshot mechanism is not supported in industrial embedded hypervisors like Jailhouse [191] or Bao [13], as it demands big capabilities of storage, i.e., memory and disk.

- **Failure modes**. Currently, the failure modes detected and thereby found during testing are generally host crashes, as they indicate a bug presence.

  However, the failure modes that can occur in a hypervisor are many, as different and complicated isolation properties should be guaranteed to the virtual machines of the guests. In ARM KVM, the first efforts were made to detect bugs in memory hierarchy management with the risk of impacting VM privacy and confidentiality.

  Hence, new efforts could be made to devise bug detectors for specific isolation properties, to be applied orthogonally to the generation of the test cases, also accepting a percentage of false positives and negatives.

5 Federated Learning

The deployment of AI and ML methods throughout many industries has been a welcome innovation that generated newfound concerns about the fairness of the results and the privacy of the involved data. Indeed, recent legislation in, e.g., Europe [79], United States [151] and China [38] have been enacted to strengthen the protection of user data used by AI and ML systems. On the other hand, companies tend to consider collected data as competing advantages and therefore are unwilling to share the data outside (sometimes even within different parts of) the organization. As a result, it is often the case that data is dispersed into many isolated islands, and ML practitioners are forbidden by laws and by legitimate owners from collecting, fusing, and ultimately using the data to improve their systems. Protecting the privacy of users and the competing advantages of companies is arguably a fair objective; nonetheless, these choices hamper the development of learning models that, by leveraging all the available

data, could make a difference in the quality of life of many people who are subjected to the decisions made using AI systems.

FL is a form of distributed learning that has been proposed by McMahan et al. [132] as a way out of this conundrum, i.e., as a way to develop better AI systems without compromising the privacy of final users and the legitimate interests of private companies. Initially deployed by Google for predicting text input on mobile devices, FL has been adopted by many other industries, such as mechanical engineering and health care [116].

FL is a learning paradigm where multiple parties (e.g., *clients*) collaborate in solving a machine learning task using their private data under the coordination of an aggregator (a.k.a. *server* or coordinator). Each client's local data is not exchanged or transferred to any participant. The learning happens in rounds where model updates are computed by clients in insulation using local and private data, then aggregated on the server, then broadcast to the clients for the next round. Although this centralized structure is by far the most used in practice, other decentralized approaches are actively being researched (for example, Swarm Learning [213]), each one with different pros and cons about performance and security [201].

There are two main federated settings: cross-device and cross-silo [98]. In *cross-device* FL [222], the parties can be edge devices (e.g., smart devices and laptops); they can be numerous (order of thousands or even millions). Parties are considered not reliable and with limited computational power. In the *cross-silo* FL setting [89], the involved parties are instead organizations; the number of parties is limited, usually in the range [2, 100]. Given the nature of the parties, it can also be assumed that communication and computation are no real bottlenecks.

FL scenarios can also be categorized according to the type of data held by each federation client. We have *horizontal* FL when the parties share a similar schema of features, but each one possesses different data samples; otherwise, we talk about *vertical* FL when each client possesses different information about the same individuals [228]. This nomenclature derives from the visual representation of how a data table would be split among the parties (the samples or individuals are the rows, and the features are the columns).

From an ML point of view, the vast majority of the FL research and industrial deployments are based on DNNs; this is due mainly to the very nature of these models, being representable as tensors, making them easy to aggregate (in simple cases an arithmetic mean is sufficient), but also for their high learning performance. However, DNNs are not the passe-partout of all ML problems, and despite their effectiveness, new approaches aiming to generalize FL to any ML model are currently being researched [161].

Federated Learning is currently a very hot research topic being deployed in a variety of settings including emerging platforms (e.g., RISC-V based systems [136]). Indeed, a number of advancements are being proposed at a very high pace. Among those, we distinguish between attempt to solve problems that are general to ML (e.g., trying to force the models to converge to fair ones [225]), and those that are specific to FL. A problem that is specific to FL

is the possibility that data are not distributed in a IID way between the clients. Indeed, Data non-IID is a huge problem in FL and many commonly employed algorithms are designed under the assumption that the data is IID among the participants of the federation, but that is usually not true. Non-IIDness may affect several aspects of the data, mainly: the quantity available to each client, the distribution of feature values, and the distribution of the labels. To overcome the issues related to non-IID distributions, a number of approaches have been proposed: in [101] control variates (variance reduction) are exploited to correct for the client-drift in its local updates; in [119] the client-drift is controlled via additional layers of batch normalization (only in local models); the key idea in [117] is to utilize the similarity between model representations to correct the local training of individual parties, i.e., conducting contrastive learning in model-level; in [118] a regularization term is added to the loss used by the clients forcing the model to be not too dissimilar to the global one; [210] proposes a general theoretical framework that allows heterogeneous number of local updates, non-IID local datasets as well as different local solvers such as GD, SGD, SGD with proximal gradients, gradient tracking, adaptive learning rates, momentum, etc..

Identified gaps. Federated Learning is mainly studied in the context of gradient descent based optimization of the training parameters. Many interesting models, ones that sports higher interpretability (e.g., decision trees), more established ways of guaranteeing fairness (e.g., SVMs), and more efficiency in learning and/or inference are, thus, not readily available for federation. In [161], a novel approach based on using AdaBoost for building the federation has been proposed, but the techniques presented therein are tailored for cross-silos federated learning and would not scale to cross-device FL due to high communication costs. More communication-efficient algorithms would be highly welcome in this space.

From a privacy stand point, FL alone is not sufficient to provide perfect protection. Model inversion attacks [74], for instance, have been proposed that allow the attacker to re-construct with high accuracy part of the datasets. Also, from the high privacy requirement in FL excludes shared infrastructures for model training, such as public cloud and HPC facilities, which creates new challenges in the deployment of the necessary HW/SW infrastructure.

6 Trustworthy AI

The increasing availability of data and widespread high-performance computing (HPC) have promoted the development of machine learning (ML) and artificial intelligence (AI) models. Machine learning models, in fact, typically require large amounts of data and computing resources for their training and optimization. Thanks to HPC and big data technologies, researchers and practitioners can more efficiently and effectively process, store, and analyze data and deploy and develop ML models. However, as the use of AI models becomes more

widespread, it is critical to assess and enforce their trustworthiness and reliability. Trustworthy AI research aims to develop AI systems that are reliable, ethical, and transparent, making AI trustworthy [113]. The critical requirements of Trustworthy AI research are explainability and transparency, robustness, fairness and diversity, and privacy. In the following, we introduce these notions, describe current approaches for their assessment and enabling, and analyze the gaps to address. We first focus on the first three requirement and then we focus on the privacy one. We first address the first three requirements (Section 6.1) and then focus on the privacy one (Sections 6.2).

6.1 Assessing and ensuring AI trustworthiness: explainability, robustness, and fairness

Given the efficiency of high-performance computing (HPC) and the increasing Big data availability, Artificial Intelligence (AI) models have increasingly higher performances. The high performance allowed their widespread adoption in multiple domains, such as medicine, law, autonomous driving, and IoT solutions. However, as AI models become increasingly pervasive, there is a growing concern about their potential negative impacts. Trustworthy AI research addresses these concerns [113]. As also highlighted by the independent High-Level Expert Group on Artificial Intelligence (AI HLEG) established by the European Commission, key requirements of Trustworthy AI are, among others, explainability and transparency, robustness and fairness and diversity. In the following, we describe current strategies to assess and enforce these requirements.

Explainability and transparency. One of the critical challenges in developing trustworthy AI is explainability. Explainable AI (XAI) refers to a set of methodologies and techniques to enable human users to understand the outcomes AI models. The task is to provide a clear and understandable explanation of how a model arrived at a particular decision or recommendation. With explainability, users can understand why the model made a decision, allowing them to decide if they trust it. However, most high-performing models are considered black boxes, not allowing the direct interpretation of their results. XAI research addresses these challenges by proposing post hoc explainability techniques that explain black box models. We can categorize explainability methods by their target, i.e., if they provide (i) global or (ii) local explanations of the model behavior [80]. Global explanations provide a global understanding of the model behavior. A common approach is to convert a black-box model into a global surrogate one that is directly interpretable [49]. Local explanations explain the motivation behind individual predictions [152, 125, 172]. Moreover, recently researchers addressed the problem from the (iii) subgroup perspective by analyzing the behavior and performance of ML models in data subgroups [42, 154, 153].

Robustness. Another key aspect of trustworthy AI is robustness. Robustness refers to the ability of an AI model to perform well under different conditions

and in the face of potential adversarial attacks or input perturbations [113]. AI models should be able to adapt to changing environments and circumstances. A relevant line of research focuses on the robustness of AI at the data level. AI models are increasingly adopted in diverse settings. They should be able to adapt to diverse data domains and distributions. Evaluating an AI model's robustness is essential to avoid disparate behavior among the data and vulnerabilities and control risks. Several techniques have been proposed for robustness test [113]. Specifically, these techniques focus on evaluating system performance along various dimensions. Recently, novel approaches have been proposed to identify data subgroups for which a model performs in an anomalous manner [42, 177, 154, 153]. For example, the methods proposed in [177, 154, 153] allow practitioners to identify the subgroups of the data for which a model performs differently than overall data. Practitioners can understand for which subgroups the model underperforms and take actions to improve the robustness and reliability of the models.

Fairness. AI models can perpetuate or exacerbate existing biases and inequalities in society. Hence, they must be designed and trained in a fair and unbiased way. Fairness in AI addresses this task from multiple perspectives [133]. From the data and training perspective, practitioners must ensure that the data used to train the system is diverse and representative. Researchers have proposed good practices for data construction and their use, such as data documentation and datasheets to report information on data creation, its characterization, and motivation [17, 16, 73]. From the algorithmic perspective, novel techniques have been proposed for designing algorithms that mitigate the risk of bias or discrimination (e.g., [7, 8, 58]). Another line of research focuses on assessing the fairness of ML models, proposing measures of fairness and bias and methods for their assessment (e.g., [83, 208, 58, 107]).

Identified gaps. Despite the recent significant steps, several gaps still need to be addressed to ensure the trustworthiness of AI. Specifically, we identified the following gaps that we aim to fill.

The first gap is benchmarking explainability methods. Given the relevance of explainability (XAI) in trustworthy AI, as outlined in Section 6.1, multiple approaches have been proposed to explain model outcomes in a human-understandable way. However, there is still a lack of easy access to using and comparing XAI explanation methods. Specifically, it is relevant to assess the quality of provided explanations, compared to their adherence to the model behavior and human reasoning. Benchmarking XAI approaches is also particularly crucial in high-risk applications such as medicine and law and social-impactful applications such as hate speech. We aim to propose novel frameworks and methodologies to benchmark XAI approaches, evaluating them in such critical applications.

The second gap is assessing and ensuring trustworthy AI in human-impactful applications such as ranking systems and law. Big Data and HPC applications

have enabled the adoption of ML models in these critical fields. Automated ranking systems are increasingly adopted for a wide range of targets, from job marketing to university applications. Ensuring these systems are reliable, ethical, and aligned with societal values is crucial. Similar considerations apply to the legal domain, which, given the advances of AI, has increased the adoption of ML decision-making support systems. We envision assessing and ensuring AI trustworthiness in these application domains, proposing ad-hoc solutions depending on the application context.

The third gap addresses the generalization of Trustworthy AI methodology to unstructured and heterogenous data as speech. Multiple of the reviewed existing techniques are specifically designed for structured (or tabular) data. The recent advantages of HPC, machine learning research, and data availability have widespread the adoption of ML models for other types of data as speech data and multi-modality. We will study and propose techniques for this context.

6.2 Models for privacy assessment in internet data

From the dawn of the Web, behavioural advertising has been a pillar of the ecosystem and entailed the collection of personal information through web tracking. Fueled by the easiness of collecting data on the Internet, behavioural advertising is built on the ability to collect and process a humongous amount of data about Internet users. Most of the technologies behind the big data and the machine learning revolution have been indeed designed to cope with the need to collect, store and process the data that internet companies have at their disposal.

This phenomenon has triggered debate and tension about data monetization and end-user privacy. Several studies measured the spread data collection [134, 60] or dug into its technical operations put in place by the companies involved [5, 173, 150]. On the other side, the implications of web tracking on users' privacy have become more and more debated by the industry [65] and by the research community [192, 130, 63]. In a nutshell, the so called web-trackers monitors and collect information about each single user (identified by means of third-party cookies) when they visit any website. This allows web-trackers to build the list of websites and webpages each user visits, and from this to extract their interests and build a single profile for each user [22]. Companies can the use this information to provide personalized ads and content in general thanks to the mechanisms of the real time bidding [226].

This fostered the birth of anti-tracking tools (i.e., the Ad and Tracker Blockers [165]), spurred the exploration of more user-friendly Personal Information Management Systems (PIMS) [93], and encouraged the legislator to issue privacy-related regulations, such as the US CCPA [27] or the European GDPR [64].

More recently, some technical solutions have appeared with the goal to balance and trade data collection and privacy. Among these, the Federated Learning of Cohorts (FLoC) has been the first public effort to go beyond the classical web tracking based on third-party cookies [168]. Proposed by Google, in FLoC users were grouped in cohorts according to the interests inferred by each one's

browser. When asking for information about a user visiting a website, third parties were offered the user's cohort, from which they could have information about the user's interests. In the intention of the proposal, FLoC provided an acceptable utility for the advertisers, while hiding the user (and thus, her identity) behind a group of peers [62]. However, criticism arose around the easiness for first- and third-party cookies to follow the user over time exploiting the sequence of cohorts to which she belongs to isolate and thus identify her [170]. The attack can exploit browser fingerprint to further improve its effectiveness [18]. FLoC's privacy anonymity properties can be broken in several ways [204]. As a response to the critics towards FLoC, Google retired the proposal and conceived the Topics API.

More recently, Google proposed the Topics API as a second proposal to mitigate the data collection while still letting behavioural advertisement. Topics API revolves around the concept of topics. Each item in a user's browsing history is mapped to a specific topic. In the current proposal, the user's browser keeps in memory the 5 most visited topics of the week, for the previous three weeks. When the user visits a website served by an advertiser, the advertiser can receive three topics from the user, one for each of the last three weeks, chosen at random among the 5 in browser's memory. In this way, the advertiser can obtain some information about the user to show her the most appropriate advertising. In a nutshell, Topics API expose users' profiles in terms of topics of interest to the websites and advertising platforms. Past works demonstrated that profiling users based on their browsing activity can present severe risks to the privacy of the users [63]. They can be identified with high probability based on the sequence of visited websites [147, 86, 207]. Mitigation such as the browser partitioned storage has been put in place to limit the risk, but ways to bypass them exist [167]. Specifically to the Topics API, a re-identification threat has already been identified by Epasto *et al.* [61] from Google. The authors carry out an information theory analysis and conclude that the attack is hardly feasible. Thomson *et al.* [200] from Mozilla have further elaborated on the conclusions by Epasto *et al.* [61], again using analytical models, and raised severe concerns on the offered privacy guarantees.

Given this scenario, we believe that studying and investigating new mechanisms for allowing users to control the information they share with online services is a perfect candidate and use case for understanding how to trade user privacy and utility of data, in a big data context, with cloud computing as the key enabling infrastructure, where computational requirement can easily grow very high.

Identified gaps. We identified some potential research gaps and open questions in the understanding of privacy and internet data:

- Balance between users' privacy and data utility: What is the trade-off between users data collection and processing and data utility? How to measure this balance?

- Understanding of privacy friendly proposals: How to validate the new proposals and FLoC or Topics API? Which eventual privacy threat can be identified? How to create analytical models that allow to gauge this trade-off?
- Data privacy and ethical concerns: How can researchers ensure they are not violating the privacy of internet users in their data collection and analysis? How can they address ethical concerns related to studying user-generated content?
- Real-time analysis: How can researchers perform real-time analysis of internet data to collect and process data? What are the technical and organizational challenges associated with this approach?

7 Social media data analysis

Social media (such as Facebook, LinkedIn, and Twitter) and instant messaging applications (such as Telegram and Whatsapp) are major forums for people to express their opinions and information, thanks to posts, groups and channels. By interacting with such applications, users build complex networks that favor the dissemination of information [9].

Social media analysis has advanced significantly in recent years, with a range of techniques and tools available for data collection and analysis from various social media platforms. Overall, the ability to collect and analyze social media data at a large scale provides valuable insights across various aspects of society.

In terms of scalable data collection from social media platforms, one approach is to use Application Programming Interfaces (APIs) provided by platforms like Twitter and Facebook to extract data [121]. However, some limitations exist with these APIs, including limited access and restrictions on data collection, making it challenging to collect data at a large scale. Another common approach to scalable data collection is to use web scraping techniques that collect data directly from the publicly available web pages of social media platforms [55]. Whereas this method presents some challenges, such as ethical and legal considerations, it allows the researcher to collect large volumes of data.

Regarding the analysis of large amounts of collected social media data in a scalable way, techniques have been developed for various social media platforms such as Facebook, Twitter, and LinkedIn (e.g., [100, 206]). In recent years, the development of graph databases and distributed systems has enabled researchers to analyze large social media graphs efficiently [155, 78].

Several academic studies have explored various methods of analyzing social media data, including network analysis [78], content analysis [187], sentiment analysis [162], and machine learning techniques [12]. These methods have been used in diverse fields, such as marketing [223, 19], politics [203], healthcare [114], and cyber-security [180].

Identified gaps. We identified some potential research gaps and open questions in scalable social media data collection and analysis:

- Quality control: How can researchers ensure the accuracy, completeness, and validity of large-scale social media data sets? How can they avoid biases in their data collection and analysis?
- Data privacy and ethical concerns: How can researchers ensure they are not violating the privacy of social media users in their data collection and analysis? How can they address ethical concerns related to studying user-generated content?
- Machine learning and AI: How can researchers leverage machine learning and AI algorithms to analyze social media data at scale? What are the limitations and challenges associated with these approaches?
- Cross-platform analysis: How can researchers integrate data from multiple social media platforms to gain a comprehensive understanding of user behavior and influence across different networks?
- Real-time analysis: How can researchers perform real-time analysis of social media data to capture events as they unfold? What are the technical and organizational challenges associated with this approach?

8 Numerical Analysis

Kernel-based schemes are popular methods used in many applied fields, such as scattered data interpolation, regression and Machine Learning. Their success both in Approximation Theory [216] and Artificial Intelligence [181] is due to the fact that they are meshfree and easy to implement in any dimension. For a complete review on the topic, we refer the reader to e.g. [25, 66, 53].

8.1 Efficient numerical software for approximation in Big Data

One of the main disadvantages of kernel-based interpolation schemes is that the matrices generated by imposing the interpolation conditions, are typically full and hence, their complexity cost is not affordable when a large number of data is available. In this setting the so-called Partition of Unity (PU) method is nowadays a well-established and efficient kernel-based interpolation scheme. First introduced in the mid 1990s, the PU method produces a global approximant by gluing together, via the use of compactly supported weights, many local fits. In recent years, PUMs have been successfully combined with a multitude of different computational methods. In particular in the approximation with radial basis functions (RBFs), the combination of RBFs with PUMs yields significantly sparser system matrices in collocation or interpolation problems, and, therefore, a considerable speed-up of calculations. Such a scheme is also

rather popular for researchers working on local collocation schemes for PDEs; refer e.g. to [33, 109].

The PU method organizes the initial set of scattered data, that lay on a multivariate domain, into several subdomains, also known as patches. Then, for each of those patches it solves a small interpolation problem. A key step in its implementation is thus the one of efficiently distributing the scattered data into the different patches. For obtaining a fast and efficient algorithm, an effective Matlab implementation of the PU scheme based on what was called the integer-based routines was proposed [29, 32]. Motivated by the growing interest of the kernel community towards Python packages for Machine Learning, a Python implementation of the PU scheme was also developed, the codes are available at https://github.com/sandro-lancellotti/PU.

Identified gaps When the local approximants in the PU scheme are radial basis functions (RBFs), several parameters have to be considered: the shape parameter in RBF, the number of patches and the radius of the subdomains in PUM. As we will see in the following, an application of PUM to the interpolation and classification of signals on graphs, has been considered and also in this context the determination of optimal parameters plays a fundamental role. In some previous papers the problem was considered from a deterministic point of view, see for instance [35, 34]. Now the goal is to exploit cross validation and likelihood estimation techniques in combination with strategies of univariate global optimization with pessimistic or optimistic improvements. Moreover, we will assume a statistical approach which can be useful also in Machine Learning and Big Data, for example considering Bayesian Optimization of the hyperparameters. In particular, Bayesian Optimisation can be used to simultaneously search the optimal values of the shape parameter and the radius in RBF interpolation together with the PUM. Since the idea is to use the algorithms in Machine Learning and Big Data applications, a parallel implementation is in progress.

8.2 Approximation and classification software for signal processing on large graphs

Very recently PUMs were combined with a local graph basis function (GBF) approximation method in order to obtain low-cost global interpolation in an efficient way on graphs. Graph signal processing is a cutting-edge research field for the study of graph signals in which mathematical processing tools as filtering, compression, noise removal,sampling, or decomposition methods are investigated [157, 194]. Graph structures appear naturally in a multitude of modern applications, as in social networks, traffic maps or biological networks. In general, these networks exhibit a large number of vertices and a highly irregular edge structure. In order to be able to deal with signals on such irregular graphs, efficient and fast processing tools are necessary. Many algorithms in graph signal processing as, for instance, the calculation of the graph Fourier transform get computationally infeasable if the size of the graph is too large or

the topological structure of the network is not sparse. PUMs allow, in an efficient way, to perform operations as signal reconstruction from samples,classification of nodes, or signal filtering locally on smaller portions of the graph, and,then, to rebuild the global signal from the local ones. This makes a PUM to an ideal auxiliary tool also if more adaptivity is required and processing steps have to be individualized to local spatial prerequisites on the graph. In fact, the split and merge procedure leads generally to a considerably lower computational cost than applying a signal processing scheme on a global scale.

In a previous paper [30], we investigated how a partition of unity can be generated efficiently on graphs and how a PUM can be combined with a local graph basis function approximation in order to obtain a low-cost global interpolation or approximation scheme. Then in [31] we presented the MATLAB package GBF-PUM that was developed and implemented to test the new scheme. In particular, we describe how the functions of the package can be used to generate a partition of unity on a graph and how signal approximation and interpolation with GBFs are implemented and combined with PUMs. This software is free and can be downloaded from the GitHub repository https://github.com/WolfgangErb/-GBFPUM. It can be useful in Graph Machine Learning applications.

Identified gaps Future work consists in providing a more adaptive technique for the selection of the partitions on the graph than the one presented in the previous papers. In particular, by using a process that automatically find an optimal number of communities based on their modularity and on the underlying graph structure, we would not need the desired number of subgraphs as an input for the GBF-PUM algorithm. Moreover, the scope is also to improve efficiency and accuracy of the scheme, by then extending its use to very large graphs and in a classification Machine Learning framework. This target will be finalized to the production of open-source software usable by the scientific community.

All the numerical experiments and tests are carried out on the infrastructure for high performance computing MathHPC, virtual cloud server of the structure HPC4AI (High-Performance Computing for Artificial Intelligence) at the University of Torino.

9 Trusted Distributed Workflows

Data management is a crucial aspect of scientific workflow orchestration. Non-functional requirements like performance optimization, privacy preservation, and security enforcement pass through a careful data management process.

Adopting the data locality principle, i.e., moving computation where the data reside, is the strategy adopted by many modern programming paradigms for Big Data analysis (e.g., MapReduce [52] and Resilient Distributed Datasets [227]) and the foundation of the federated learning approach [131]. Avoiding data transfers means removing related communication overhead and security risks and guaranteeing data privacy and integrity without trusting third-party

computational resources.

Nevertheless, in some cases, data transfers are worth or even unavoidable. Modern deep neural networks, large-scale genomics, and high-fidelity digital twins are all examples of computations that require computing power and communication speed that only the largest High-Performance Computing centers in the world can offer. However, they also need input datasets commonly generated, analyzed, and pre-processed elsewhere, from cloud resources to edge devices. Also, directly uploading such data to a shared HPC resource is often impossible, either because they are sensitive data protected by privacy rights or because they constitute a strategic value for the data owner.

Therefore, an educated application of the privacy-by-design and security-by-design principles is a fundamental requirement for scientific workflows. However, leaving this responsibility entirely in the hands of the users is not an option, as effectively dealing with security-related aspects is not trivial for domain experts without a strong Computer Science background. A better approach is to move security and privacy implementation at the workflow management system level, requiring users to define trust perimeters and security levels in the information flow [54, 88].

Developing portable solutions for end-to-end trust in distributed workflows is a pivotal research field. One of the most critical aspects is to guarantee data security and privacy at rest, i.e., when they are persisted in remote untrusted storage, in a completely transparent way to the host application. A possible strategy is to combine a secure key generation mechanism with a file system encryption library.

The Laniakea workflow platform [197] uses the PBKDF2 key derivation function to generate a master key for each user automatically and stores it in a Hashicorp® Vault[2] instance. Each data volume is then encrypted using the Linux Unified Key Setup (LUKS) library and a volume-specific encryption key stored in Vault and bound to the user's master key through a one-time authentication token.

Secure HPC [145, 146] relies on asymmetric key pairs for integrity checks and encryption, Vault as a key management system, and a combination of LUKS and Singularity containers for data encryption. Plus, it uses advanced features of the Intel® OmniPath to partition the compute nodes at the network level, enforcing multi-tenancy.

The WfExS-backend [67] relies on Crypt4GH [185] to exchange the encryption key of a FUSE encrypted storage (EncFS[3] or gocryptfs[4]). Using FUSE enhances portability, as it does not require kernel-level support.

Identified gaps. All these approaches can protect data at rest from other users of remote computing resources, but privileged users and infrastructure administrators are always included in the trust perimeter. Indeed, all the data

[2]https://www.vaultproject.io/
[3]https://vgough.github.io/encfs/
[4]https://nuetzlich.net/gocryptfs/

live in the process memory in unencrypted form during computation, allowing privileged users to access private information.

Enabling zero-trust distributed workflows is the natural evolution of these approaches. Secure enclaves and remote attestation mechanisms built on top of them are promising approaches to end-to-end trusted workflows. They can implement a fully protected key exchange mechanism for data encryption at rest and prevent data disclosure during execution.

A direct interaction between application code and enclave APIs guarantees the best performance vs security trade-off, as the enclave can protect only code sections that deal with sensitive data. Still, this approach requires detailed knowledge of the application's information flow and the enclave's internal details, increases complexity, and hinders the portability and maintainability of the application code.

Library operating systems, such as Gramine [202] and Occlum [188], can execute unmodified applications under complete enclave protection. Moreover, by wrapping and checking system calls, they shield the application from additional classes of attacks, such as Iago attacks [36]. Users must configure some enclave parameters in an external, declarative manifest file. Still, a proper configuration of these manifests is non-trivial for domain experts, and both security and performance tend to be very sensitive to misconfigurations.

10 Stochastics Models

Stochastic models may become useful tools to support AI system development and study. In particular they can play an important role in some industrial applications when the interest focus on optimal choices requested, for example, in predictive maintenance or reliability problems. In this context, optimal stopping methods give alternative approach with respect to the use of machine learning or advanced statistical methods [ref]. Optimal stopping methods include all the methodologies aiming to select the best time when to stop a procedure in order to minimize a cost function. Often this implies to look for the time that minimizes the probability of false alarm and the expected delay. Presently, optimal stopping tools are available for specific stochastic processes such as, for example Poisson process, Brownian motion, Bessel process. [ref].

Furthermore, in some instances reliability problems involve first passage time problems. In such problems one looks for the first time in which the observed process reaches a given level or the first time when the process exit from a strip. Numerical and simulation tools are available in the case of one dimensional stochastic processes [] such as diffusion or specific Levy processes. []

Another approach connected with first passage time and useful for reliability make use of the inverse first passage time problem. Here one assume that the process and the fist passage times are known and looks for the shape of the boundary. Inverse first passage time methods have been used as an alternative tool to classify observed data by comparing the boundaries corresponding to different sets of data [].

An important problem that arises for any modelling attempt consists in accounting for dependencies between involved variables. The standard technique is based on correlation analysis but it highlights linear dependencies between variables. A useful tool in this context is given by copulas []. These are multivariate functions that allow to detect dependencies between random variables separating the joint and the marginal behaviour.

Identified gaps. All the proposed techniques should be adapted to the specific instances of interest. In particular, industrial applications request the development of stopping time methods for processes different from the theoretical ones discussed in the recent mathematical literature. Approximations of real processes to theoretical ones will be necessary. As far as first passage time problems are concerned, there are open questions for processes of interest in reliability studies or for queuing modeling. In particular, the first passage time for a Lindley process is not known (such process is a random walk, constrained to be positive and characterized by jumps of continuous size, Laplace distributed). Furthermore, exact algorithms for inverse first passage time should be developed for specific processes of applied interest. In presence of specific problems suggested by applications a copula approach can be investigated adapting the existing mathematics to the specific instances.

11 Gap analysis

Below we summarize the results of the background and gap analysis carried out within Flagship 4.

Gap#	**Related area**	**Gap description**
Gap01	RISC-V TEEs	limited TEE support in RISC-V microcontroller-class systems
Gap02	RISC-V TEEs	limited privilege management flexibility in RISC-V microcontroller-class systems
Gap03	Accelerator TEE support	CPU-based TEEs unfit for HPC/big data
Gap04	Accelerator TEE support	no commercial solutions accelerator-based TEEs
Gap05	Secure virtualization	limited support for RPU virtualization
Gap06	Secure virtualization	limited hypervisor misbehavior detection in hardware-assisted virtualization
Gap07	Secure virtualization	difficulty of test cases for tracing CPU virtualization bugs
Gap08	Secure virtualization	trade-offs between bug detection and isolation properties

Gap09	Federated Learning	FL lacks approaches enabling the usage of ML models that are impossible (or not easy) to be trained without gradient descent
Gap10	Federated Learning	FL needs to be augmented with additional tools for provide better privacy guarantees
Gap11	Federated Learning	Privacy constraints prevent using standard HW/SW platform as deployment target
Gap12	Trustworthy AI	lack of AI explainability benchmarking methods
Gap13	Trustworthy AI	lack of trustworthy AI solutions for ranking systems and law
Gap14	Trustworthy AI	limited support for trustworthy AI with unstructured and heterogenous data
Gap15	Trustworthy AI	user privacy vs. data utility tradeoffs
Gap16	Trustworthy AI	lack of analytical models for validating trustworthy AI proposals
Gap17	Trustworthy AI	ethical concerns in data collection
Gap18	Trustworthy AI	limited real-time analysis support for internet data
Gap19	Social media data analysis	limited means to assess accuracy, completeness, and validity of large-scale social media data sets
Gap20	Social media data analysis	data privacy and ethical concerns for privacy violation in social media data collection and analysis
Gap21	Social media data analysis	need to improved AI algorithms for analyzing social media data at scale
Gap22	Social media data analysis	need for ways of integrating cross-platform data from multiple social media
Gap23	Social media data analysis	lack of support for real-time analysis of social media data
Gap24	Numerical Analysis	NA needs efficient numerical SW for approximation using hyperparameters tuning in Big Data
Gap25	Numerical Analysis	NA lacks software for approximation and classification on large graphs
Gap26	Trusted Distributed Workflows	need for zero-trust distributed workflows
Gap27	Trusted Distributed Workflows	complexity, limited portability and maintainability of enclave-based application code
Gap28	Stochastics Models	gap and approximations between real processes and theoretical processes

Gap29	Stochastics Models	need for exact algorithms for inverse first passage time
Gap30	Stochastics Models	first passage time for a Lindley process not known

Table 2: Flagship 4 gap analysis.

References

[1] BlackBerry Limited., Are Hypervisors the Answer to the Coming silicon Shortages? `https://blackberry.qnx.com/content/dam/blackberry-com/Documents/pdf/BlackBerry_QNX_Hypervisor_WhitePaper_22April2021_FINAL.pdf`.

[2] Laure Abdallah, Mathieu Jan, Jérôme Ermont, and Christian Fraboul. Reducing the contention experienced by real-time core-to-i/o flows over a tilera-like network on chip. In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 86–96. IEEE, 2016.

[3] Luca Abeni and Dario Faggioli. Using Xen and KVM as real-time hypervisors. *Journal of Systems Architecture*, 106:101709, 2020.

[4] Darren Abramson, Jeff Jackson, Sridhar Muthrasanallur, Gil Neiger, Greg Regnier, Rajesh Sankaran, Ioannis Schoinas, Rich Uhlig, Balaji Vembu, and John Wiegert. Intel Virtualization Technology for Directed I/O. *Intel technology journal*, 10(3), 2006.

[5] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 674–689, 2014.

[6] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. *ACM Sigplan Notices*, 41(11):2–13, 2006.

[7] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69. PMLR, 2018.

[8] Alekh Agarwal, Miroslav Dudík, and Zhiwei Steven Wu. Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning*, pages 120–129. PMLR, 2019.

[9] Mohammed Ali Al-Garadi, Kasturi Dewi Varathan, Sri Devi Ravana, Ejaz Ahmed, Ghulam Mujtaba, Muhammad Usman Shahid Khan, and

Samee U Khan. Analysis of online social network connections for identification of influential users: Survey and open research issues. *ACM Computing Surveys*, 51(1):1–37, 2018.

[10] AMD Technology. AMD64 Architecture Programmer's Manual Volume 2: System Programming.

[11] Nadav Amit, Dan Tsafrir, Assaf Schuster, Ahmad Ayoub, and Eran Shlomo. Virtual CPU validation. *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[12] TK Balaji, Chandra Sekhara Rao Annavarapu, and Annushree Bablani. Machine learning algorithms for social media analysis: A survey. *Computer Science Review*, 40:100395, 2021.

[13] N. N. Bao et al. A Real-Time Disruption Prediction Tool for VDE on EAST. *IEEE Trans. Plasma Sci.*, 48(3):715–720, 2020.

[14] Richard Barry et al. Freertos. *Internet, Oct*, 2008.

[15] Felix Baum and Arvind Raghuraman. Making full use of emerging ARM-based heterogeneous multicore SoCs. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[16] Emily M Bender and Batya Friedman. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 2018.

[17] Misha Benjamin, Paul Gagnon, Negar Rostamzadeh, Chris Pal, Yoshua Bengio, and Alex Shee. Towards standardization of data licenses: The montreal data license. *arXiv preprint arXiv:1903.12262*, 2019.

[18] Alex Berke and Dan Calacci. Privacy limitations of interest-based advertising on the web: A post-mortem empirical analysis of google's floc. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 337–349, New York, NY, USA, 2022. Association for Computing Machinery.

[19] Fabio Bertone, Luca Vassio, and Martino Trevisan. The stock exchange of influencers: A financial approach for studying fanbase variation trends. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '21, page 431–435, New York, NY, USA, 2021. Association for Computing Machinery.

[20] Alessandro Biondi, Daniel Casini, Giorgiomaria Cicero, Niccolò Borgioli, Giorgio Buttazzo, Gaetano Patti, Luca Leonardi, Lucia Lo Bello, Marco Solieri, Paolo Burgio, et al. Sphere: A multi-soc architecture for next-generation cyber-physical systems based on heterogeneous platforms. *IEEE Access*, 9:75446–75459, 2021.

[21] Christophe Bobda et al. The Future of FPGA Acceleration in Datacenters and the Cloud. *ACM Trans. Reconfigurable Technology and Systems*, 15(3):1–42, 2022.

[22] Sophie C Boerman, Sanne Kruikemeier, and Frederik J Zuiderveen Borgesius. Online behavioral advertising: A literature review and research agenda. *Journal of advertising*, 46(3):363–376, 2017.

[23] Michael Brengel, Michael Backes, and Christian Rossow. Detecting hardware-assisted virtualization. In *International Conference on Detection of intrusions and malware, and vulnerability assessment*, 2016.

[24] Thomas C. Bressoud and Fred B. Schneider. Hypervisor-based fault tolerance. In *TOCS*, 1996.

[25] Martin D Buhmann. *Radial Basis Functions: Theory and Implementation* . Cambridge Univ. Press, 2003.

[26] Alexander Bulekov, Bandan Das, Stefan Hajnoczi, and Manuel Egele. Morphuzz: Bending (Input) Space to Fuzz Virtual Devices. In *USENIX Security Symposium*, 2022.

[27] California State Legislature. California Consumer Privacy Act of 2018. `https://oag.ca.gov/privacy/ccpa` (Last accessed February 27, 2023), 2018.

[28] Daniel Casini, Alessandro Biondi, Giorgiomaria Cicero, and Giorgio Buttazzo. Latency analysis of i/o virtualization techniques in hypervisor-based real-time systems. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 306–319. IEEE, 2021.

[29] Roberto Cavoretto and Alessandra De Rossi. A trivariate interpolation algorithm using a cube-partition searching procedure. *SIAM J. Sci. Comput.*, 37:A1891–A1908, 2015.

[30] Roberto Cavoretto, Alessandra De Rossi, and Wolfgang Erb. Partition of unity methods for signal processing on graphs. *J. Fourier Anal. Appl.*, 27(66), 2021.

[31] Roberto Cavoretto, Alessandra De Rossi, and Wolfgang Erb. Gbfpum - a matlab package for partition of unity based signal interpolation and approximation on graphs. *Dolomites Res. Notes Approx*, 15:25–34, 2022.

[32] Roberto Cavoretto, Alessandra De Rossi, Sandro Lancellotti, and Emma Perracchione. Software implementation of the partition of unity method. *Dolomites Res. Notes Approx.*, 74:35–46, 2022.

[33] Roberto Cavoretto, Alessandra De Rossi, Mukhametzhanov S Marat, and Yaroslav D. Sergeyev. Error indicators and refinement strategies for solving poisson problems through a rbf partition of unity collocation scheme. *Appl. Math. Comput.*, 369(124824):305–327, 2020.

[34] Roberto Cavoretto, Alessandra De Rossi, Marat S Mukhametzhanov, and Yaroslav D Sergeyev. On the search of the shape parameter in radial basis functions using univariate global optimization methods. *J. Global Optim.*, 79:305–327, 2021.

[35] Roberto Cavoretto, Alessandra De Rossi, and Emma Perracchione. Optimal selection of local approximants in rbf-pu interpolation. *J. Sci. Comput.*, 74:1–22, 2018.

[36] Stephen Checkoway and Hovav Shacham. Iago attacks: why the system call API is a bad untrusted RPC interface. In Vivek Sarkar and Rastislav Bodík, editors, *Architectural Support for Programming Languages and Operating Systems, ASPLOS 2013, Houston, TX, USA, March 16-20, 2013*, pages 253–264. ACM, 2013.

[37] Daming D Chen, Maverick Woo, David Brumley, and Manuel Egele. Towards automated dynamic analysis for linux-based embedded firmware. In *NDSS*, volume 1, pages 1–1, 2016.

[38] Jihong Chen and Jiabin Sun. Understanding the chinese data security law. *International Cybersecurity Law Review*, 2(2):209–221, 2021.

[39] Liqun Chen, Siani Pearson, and Athanasios Vamvakas. On enhancing biometric authentication with data protection. In *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No. 00TH8516)*, volume 1, pages 249–252. IEEE, 2000.

[40] Jung Hee Cheon, Miran Kim, and Kristin Lauter. Homomorphic computation of edit distance. In Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 194–212, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[41] Jim Chow, Tal Garfinkel, and Peter M. Chen. VMwareDecoupling Dynamic Program Analysis from Execution in Virtual Environments. In *USENIX Annual Technical Conference*, 2008.

[42] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1550–1553. IEEE, 2019.

[43] Alessandro Cilardo, Marcello Cinque, Luigi De Simone, and Nicola Mazzocca. Virtualization over Multiprocessor System-on-Chip: an Enabling Paradigm for Industrial IoT. *arXiv preprint arXiv:2112.15404*, 2021.

[44] Marcello Cinque, Domenico Cotroneo, Luigi De Simone, and Stefano Rosiello. Virtualizing mixed-criticality systems: A survey on industrial trends and issues. *Future Generation Computer Systems*, 2021.

[45] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[46] Victor Costan and Srinivas Devadas. Intel SGX explained, 4.2 Arm TrustZone. *IACR Cryptol. ePrint Arch.*, 2016(86), 2016.

[47] 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, 2016.

[48] Cox and J. Delvare. *dmidecode(8) - Linux man page. die.net. Retrieved June 17, 2017.* Last accessed:12/22.

[49] Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8, 1995.

[50] C. Dall, S. Li, J. T. Lim, J. Nieh, and G. Koloventzos. ARM virtualization: performance and architectural implications. In *Annual International Symposium on Computer Architecture*, pages 304–316. IEEE, 2016.

[51] Asmit De, Aditya Basu, Swaroop Ghosh, and Trent Jaeger. FIXER: Flow integrity extensions for embedded RISC-V. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 348–353. IEEE, 2019.

[52] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *CACM*, 51(1):107–113, 2008.

[53] Marc P Deisenroth, Aldo A Faisal, and Cheng S Ong. *Mathematics for Machine Learning* . Cambridge Univ. Press, 2020.

[54] Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.

[55] Lusiana Citra Dewi, Alvin Chandra, et al. Social media web scraping using social media developers api and regex. *Procedia Computer Science*, 157:444–449, 2019.

[56] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza A. Basrai, and Peter M. Chen. ReVirt: enabling intrusion analysis through virtual-machine logging and replay. In *OPSR*, 2002.

[57] George W. Dunlap, Dominic G. Lucchetti, Michael A. Fetterman, and Peter M. Chen. Execution replay of multiprocessor virtual machines. In *VEE '08*, 2008.

[58] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[59] Kevin Elphinstone and Gernot Heiser. From l3 to sel4 what have we learnt in 20 years of l4 microkernels? In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 133–150, 2013.

[60] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1388–1401, 2016.

[61] Alessandro Epasto, Andres Munoz Medina, Christina Ilvento, and Josh Karlin. Measures of cross-site re-identification risk: An analysis of the topics api proposal. `https://github.com/patcg-individual-drafts/topics/blob/main/topics_analysis.pdf` (Last accessed February 27, 2023), 2022.

[62] Alessandro Epasto, Andrés Muñoz Medina, Steven Avery, Yijian Bai, Robert Busa-Fekete, CJ Carey, Ya Gao, David Guthrie, Subham Ghosh, James Ioannidis, Junyi Jiao, Jakub Lacki, Jason Lee, Arne Mauser, Brian Milch, Vahab Mirrokni, Deepak Ravichandran, Wei Shi, Max Spero, Yunting Sun, Umar Syed, Sergei Vassilvitskii, and Shuo Wang. Clustering for private interest-based advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2802–2810, New York, NY, USA, 2021. Association for Computing Machinery.

[63] José Estrada-Jiménez, Javier Parra-Arnau, Ana Rodríguez-Hoyos, and Jordi Forné. Online advertising: Analysis of privacy threats and protection approaches. *Computer Communications*, 100:32–51, 2017.

[64] European Parliament and Council of European Union. Directive 95/46/EC. General Data Protection Regulation. `http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf` (Last accessed February 27, 2023), 2016.

[65] Stephen Farrell and Hannes Tschofenig. Pervasive Monitoring Is an Attack. RFC 7258, May 2014.

[66] Gregory Fasshauer and Michael McCourt. *Kernel-based Approximation Methods using Matlab* . World Scientific, 2015.

[67] José María Fernández, Laura Rodríguez-Navas, and Adrián Muñoz-Cívico. WfExS-backend, 6 2022.

[68] Christopher W Fletcher, Marten van Dijk, and Srinivas Devadas. A secure processor architecture for encrypted computation on untrusted programs. In *Proceedings of the seventh ACM workshop on Scalable trusted computing*, pages 3–8, 2012.

[69] Pedro Fonseca, Xi Wang, and Arvind Krishnamurthy. MultiNyx: a multi-level abstraction framework for systematic analysis of hypervisors. *Proceedings of the Thirteenth EuroSys Conference*, 2018.

[70] Gautam Gala, Gerhard Fohler, Peter Tummeltshammer, Stefan Resch, and Reingard Hametner. RT-cloud: Virtualization technologies and cloud computing for railway use-case. In *24th IEEE International Symposium On Real-Time distributed Computing (IEEE ISORC)*. IEEE, 2021.

[71] Tal Garfinkel, Keith Adams, Andrew Warfield, and Jason Franklin. Compatibility is not transparency: Vmm detection myths and realities. In *HotOS*, 2007.

[72] Xinyang Ge, Ben Niu, Robert Brotzman, Yaohui Chen, HyungSeok Han, Patrice Godefroid, and Weidong Cui. HyperFuzzer: An Efficient Hybrid Fuzzer for Virtual CPUs. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[73] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[74] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[75] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs. *Journal of Cryptographic Engineering*, 5(2):95–112, 2015.

[76] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[77] Dieter Gollmann. Computer security. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):544–554, 2010.

[78] Carlos Henrique Gomes Ferreira, Fabricio Murai, Ana PC Silva, Martino Trevisan, Luca Vassio, Idilio Drago, Marco Mellia, and Jussara M Almeida. On network backbone extraction for modeling online collective behavior. *Plos one*, 17(9):e0274218, 2022.

[79] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct 2017.

[80] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[81] Prasanna Hambarde, Rachit Varma, and Shivani Jha. The survey of real time operating system: Rtos. In *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, pages 34–39. IEEE, 2014.

[82] Reinhard Hametner, Peter Tummeltshammer, Stefan Resch, and Wolfgang Wernhar. Cloud architecture for sil4 railway applications. *Thales Austria GmbH, Signal + Draht - SIGNALLING & DATACOMMUNICATION magazine, Eurailpress Archiv - DVV Media Group GmbH*, March 2022.

[83] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[84] Yacine Hebbal, Sylvie Laniepce, and Jean-Marc Menaud. Virtual Machine Introspection: Techniques and Applications. *2015 10th International Conference on Availability, Reliability and Security*, pages 676–685, 2015.

[85] Andrew Henderson, Heng Yin, Guang yao Jin, Hao Han, and Hongmei Deng. VDF: Targeted Evolutionary Fuzz Testing of Virtual Devices. In *RAID*, 2017.

[86] Dominik Herrmann, Christian Banse, and Hannes Federrath. Behavior-based tracking: Exploiting characteristic patterns in dns traffic. *Computers & Security*, 39:17–33, 2013.

[87] Cheol-Ho Hong, Ivor Spence, and Dimitrios S Nikolopoulos. GPU virtualization and scheduling methods: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 50(3):1–37, 2017.

[88] Wei-kuang Huang and Vijayalakshmi Atluri. Secureflow: A secure web-enabled workflow management system. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control, RBAC 1999, Fairfax, VA, USA, October 28-29, 1999*, pages 83–94, 1999.

[89] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7865–7873, 2021.

[90] CO Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3 (3A, 3B, 3C & 3D): System Programming Guide. *Denver,[2006]*, 2022.

[91] ISO. Road vehicles – Functional safety, 2011.

[92] Benedikt Janßen et al. Towards a type 0 hypervisor for dynamic reconfigurable systems. In *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–7, 2017.

[93] Nikhil Jha, Martino Trevisan, Luca Vassio, Marco Mellia, Stefano Traverso, Alvaro Garcia-Recuero, Nikolaos Laoutaris, Amir Mehrjoo, Santiago Andrés Azcoitia, Ruben Cuevas Rumin, et al. A pims development kit for new personal data platforms. *IEEE Internet Computing*, 26(3):79–84, 2022.

[94] Somesh Jha, Louis Kruger, and Vitaly Shmatikov. Towards practical privacy for genomic computation. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 216–230, 2008.

[95] Zhe Jiang et al. BlueVisor: Time-Predictable Hardware Hypervisor for Many-core Embedded Systems. *IEEE Trans. Computers*, 2021.

[96] Zhe Jiang, Kecheng Yang, Yunfeng Ma, Nathan Fisher, Neil Audsley, and Zheng Dong. I/o-guard: Hardware/software co-design for i/o virtualization with guaranteed real-time performance. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1159–1164. IEEE, 2021.

[97] Ashlesha Joshi, Samuel T. King, George W. Dunlap, and Peter M. Chen. Detecting past and present intrusions through vulnerability-specific predicates. In *Symposium on Operating Systems Principles*, 2005.

[98] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[99] Chia Hung Kao. Survey on Evaluation of IoT Services Leveraging Virtualization Technology. In *Proc. 2020 5th International Conference on Cloud Computing and Internet of Things*, pages 26–34, 2020.

[100] Maria Karanasou, Anneta Ampla, Christos Doulkeridis, and Maria Halkidi. Scalable and real-time sentiment analysis of twitter data. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 944–951. IEEE, 2016.

[101] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[102] Namhoon Kim, Stephen Tang, Nathan Otterness, James H Anderson, F Donelson Smith, and Donald E Porter. Supporting i/o and ipc via fine-grained os isolation for mixed-criticality real-time tasks. In *Proceedings of the 26th international conference on real-time networks and systems*, pages 191–201, 2018.

[103] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, et al. seL4: Formal verification of an OS kernel. In *Symposium on Operating systems principles*, pages 207–220, 2009.

[104] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[105] Bastian Koppelmann, Peer Adelt, Wolfgang Mueller, and Christoph Scheytt. RISC-V extensions for bit manipulation instructions. In *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 41–48. IEEE, 2019.

[106] Nick Kossifidis, Joe Xie, Bill Huffman, Allen Baum, Greg Favor, Tariq Kurd, and Fumio Arakawa. Epmp: PMP Enhancements for memory access and execution prevention on Machine mode (Smepmp). 2021.

[107] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *Advances in neural information processing systems*, 30, 2017.

[108] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[109] Elisabeth Larsson, Victor Shcherbakov, and Alfa Heryudono. A least squares radial basis function partition of unity method for solving PDEs. *SIAM J. Sci. Comput.*, 39:A2538–A2563, 2017.

[110] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16, 2020.

[111] Yunsup Lee. RISC-V "Rocket chip" SoC generator in Chisel. In *Online slides:" https://riscv. org/wp-content/uploads/2015/01/riscv-rocket-chip-generator-workshop-jan2015. pdf*, 2015.

[112] Marianna Lezzi, Mariangela Lazoi, and Angelo Corallo. Cybersecurity for Industry 4.0 in the current literature: A reference framework. *Computers in Industry*, 103:97–110, 2018.

[113] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *ACM Comput. Surv.*, 55(9), jan 2023.

[114] Cuilian Li, Li Jia Chen, Xueyu Chen, Mingzhi Zhang, Chi Pui Pang, and Haoyu Chen. Retrospective analysis of the possibility of predicting the covid-19 outbreak from internet searches and social media data, china, 2020. *Eurosurveillance*, 25(10):2000199, 2020.

[115] Hao Li, Xuefei Xu, Jinkui Ren, and Yaozu Dong. ACRN: a big little hypervisor for IoT development. In *Proc. 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 31–44, 2019.

[116] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[117] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021.

[118] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[119] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

[120] Linfo. *The dmesg command. The linux information project, 2007. (12 January 2007).* Last accessed:12/22.

[121] Stine Lomborg and Anja Bechmann. Using apis for data collection on social media. *The Information Society*, 30(4):256–265, 2014.

[122] LOWRisc. Ibex: An embedded 32 bit RISC-V CPU core. 2021.

[123] Enno Lubbers and Marco Platzner. Reconos: An rtos supporting hard- and software threads. In *2007 International Conference on Field Programmable Logic and Applications*, pages 441–446. IEEE, 2007.

[124] Pierre Lucas, Kevin Chappuis, Michele Paolino, Nicolas Dagieu, and Daniel Raho. VOSYSmonitor, a Low Latency Monitor Layer for Mixed-Criticality Systems on ARMv8-A. In *29th Euromicro Conf. Real-Time Systems (ECRTS 2017)*, pages 6:1–6:18, 2017.

[125] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[126] Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Krste Asanovic, John Kubiatowicz, and Dawn Song. Phantom: Practical oblivious computation in a secure processor. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 311–324, 2013.

[127] Mailing list. Xen Memory De-duplication.

[128] José Martins, Adriano Tavares, Marco Solieri, Marko Bertogna, and Sandro Pinto. Bao: A lightweight static partitioning hypervisor for modern multi-core embedded systems. In *Workshop on Next Generation Real-Time Embedded Systems (NG-RES 2020)*, pages 3:1–3:14, 2020.

[129] Miguel Masmano, Ismael Ripoll, Alfons Crespo, and J Metge. Xtratum: a hypervisor for safety critical embedded systems. In *11th Real-Time Linux Workshop*, pages 263–272, 2009.

[130] Jonathan R Mayer and John C Mitchell. Third-party web tracking: Policy and technology. In *2012 IEEE symposium on security and privacy*, pages 413–427. IEEE, 2012.

[131] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282, 2017.

[132] McMahan et al. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[133] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[134] Hassan Metwalley, Stefano Traverso, Marco Mellia, Stanislav Miskovic, and Mario Baldi. The online tracking horde: a view from passive measurements. In *International Workshop on Traffic Monitoring and Analysis*, pages 111–125. Springer, 2015.

[135] Chris Mitchell. *Trusted computing*, volume 6. Iet, 2005.

[136] Gianluca Mittone, Nicolò Tonci, Robert Birke, Iacopo Colonnelli, Doriana Medić, Andrea Bartolini, Roberto Esposito, Emanuele Parisi, Francesco Beneventi, Mirko Polato, Massimo Torquati, Luca Benini, and Marco Aldinucci. Experimenting with emerging risc-v systems for decentralised machine learning, 2023.

[137] Carlos Moratelli, Sergio Johann, Marcelo Neves, and Fabiano Hessel. Embedded virtualization for the design of secure IoT applications. In *2016 International Symposium on Rapid System Prototyping (RSP)*, pages 1–5, 2016.

[138] Tiago Mück, Antonio A Fröhlich, Giovani Gracioli, Amir M Rahmani, João Gabriel Reis, and Nikil Dutt. Chips-ahoy: A predictable holistic cyber-physical hypervisor for mpsocs. In *Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 73–80, 2018.

[139] Cheolwoo Myung, Gwangmu Lee, and Byoungyoung Lee. MundoFuzz: Hypervisor Fuzzing with Statistical Coverage Testing and Grammar Inference. In *USENIX Security Symposium*, 2022.

[140] Gil Neiger, Amy Santoni, Felix Leung, Dion Rodgers, and Rich Uhlig. Intel virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal*, 10(3), 2006.

[141] Gil Neiger, Amy Santoni, Felix Leung, Dion Rodgers, and Rich Uhlig. Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. *Intel Technology Journal*, 10(3), 2006.

[142] NIST. CVE-2010-0435.

[143] NIST. CVE-2011-1936.

[144] NIST. CVE-2020-2732.

[145] Hendrik Nolte, Simon Hernan Sarmiento Sabater, Tim Ehlers, and Julian Kunkel. A secure workflow for shared HPC systems. In *22nd IEEE International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2022, Taormina, Italy, May 16-19, 2022*, pages 965–974. IEEE, 2022.

[146] Hendrik Nolte, Nicolai Spicher, Andrew Russel, Tim Ehlers, Sebastian Krey, Dagmar Krefting, and Julian Kunkel. Secure HPC: A workflow providing a secure partition on an HPC system. *Future Generation Computer Systems*, 141:677–691, 2023.

[147] Lukasz Olejnik, Claude Castelluccia, and Artur Janc. Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns. In *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*, Spain, July 2012.

[148] Online. *Digitale Schiene Deutschland*. Last accessed:12/22.

[149] Gaoning Pan, Xingwei Lin, Xuhong Zhang, Yongkang Jia, Shouling Ji, Chunming Wu, Xinlei Ying, Jiashui Wang, and Yanjun Wu. V-Shuttle: Scalable and Semantics-Aware Hypervisor Virtual Device Fuzzing. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[150] Emmanouil Papadogiannakis, Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. *User Tracking in the Post-Cookie Era: How Websites Bypass GDPR Consent to Track Users*, page 2130–2141. Association for Computing Machinery, New York, NY, USA, 2021.

[151] Stuart L Pardau. The california consumer privacy act: towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y*, 23:68, 2018.

[152] Eliana Pastor and Elena Baralis. Explaining black box models by means of local rules. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 510–517, New York, NY, USA, 2019. Association for Computing Machinery.

[153] Eliana Pastor, Luca De Alfaro, and Elena Baralis. Looking for trouble: Analyzing classifier behavior via pattern divergence. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1400–1412, 2021.

[154] Eliana Pastor, Andrew Gavgavian, Elena Baralis, and Luca de Alfaro. How divergent is your data? *Proceedings of the VLDB Endowment*, 14(12):2835–2838, 2021.

[155] NS Patil, P Kiran, NP Kiran, and Naresh Patel KM. A survey on graph database management techniques for huge unstructured data. *International Journal of Electrical and Computer Engineering*, 8(2):1140, 2018.

[156] Nathan Pemberton, Anton Zabreyko, Zhoujie Ding, Randy Katz, and Joseph Gonzalez. Kernel-as-a-service: A serverless interface to gpus. *arXiv preprint arXiv:2212.08146*, 2022.

[157] Isaac Z Pesenson and Meyer Z Pesenson. Graph signal sampling and interpolation based on clusters and averages. *J. Fourier Anal. Appl.*, 27(39):A2538–A2563, 2021.

[158] Charles P Pfleeger. *Security in computing*. Prentice-Hall, Inc., 1988.

[159] S. Pinto and N. Santos. Demystifying arm trustzone: A comprehensive survey. *Computing Surveys*, 51:1–36, 2019.

[160] Sanndro Pinto, Hugo Araujo, Daniel Oliveira, Jose Martins, and Adriano Tavares. Virtualization on trustzone-enabled microcontrollers? voilà! In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 293–304. IEEE, 2019.

[161] Mirko Polato, Roberto Esposito, and Marco Aldinucci. Boosting the federation: Cross-silo federated learning without gradient descent. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2022.

[162] Federico Alberto Pozzi, Elisabetta Fersini, Enza Messina, and Bing Liu. Challenges of sentiment analysis in social networks: an overview. *Sentiment analysis in social networks*, pages 1–11, 2017.

[163] ARM Processors. ARM TrustZone on Cortex-M23 and Cortex-M33. 2016.

[164] ARM Processors. ARM TrustZone. 2019.

[165] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. Annoyed users: Ads and ad-block usage in the wild. In *Proceedings of the 2015 Internet Measurement Conference*, pages 93–106, 2015.

[166] Ralf Ramsauer, Jan Kiszka, Daniel Lohmann, and Wolfgang Mauerer. Look mum, no VM exits!(almost). *arXiv preprint arXiv:1705.06932*, 2017.

[167] Audrey Randall, Peter Snyder, Alisha Ukani, Alex C. Snoeren, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. Measuring uid smuggling in the wild. In *Proceedings of the 22nd ACM Internet Measurement Conference*, page 230–243, New York, NY, USA, 2022. Association for Computing Machinery.

[168] Deepak Ravichandran and S Vasilvitskii. Evaluation of cohort algorithms for the floc api. *Google Research & Ads white paper*, 2021.

[169] Ling Ren, Christopher W Fletcher, Albert Kwon, Marten Van Dijk, and Srinivas Devadas. Design and implementation of the ascend secure processor. *IEEE Transactions on Dependable and Secure Computing*, 16(2):204–216, 2017.

[170] Eric Rescorla and Martin Thomson. Technical comments on floc privacy. 2021.

[171] Francesco Restuccia, Alessandro Biondi, Mauro Marinoni, Giorgiomaria Cicero, and Giorgio Buttazzo. Axi hyperconnect: A predictable, hypervisor-level interconnect for hardware accelerators in fpga soc. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[172] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[173] Valentino Rizzo, Stefano Traverso, and Marco Mellia. Unveiling web fingerprinting in the wild via code mining and machine learning. *Proceedings on Privacy Enhancing Technologies*, 2021(1):43–63, 2021.

[174] Shahin Roozkhosh, Denis Hoornaert, and Renato Mancuso. Caesar: Coherence-aided elective and seamless alternative routing via on-chip fpga. In *2022 IEEE Real-Time Systems Symposium (RTSS)*, pages 356–369. IEEE, 2022.

[175] Shahin Roozkhosh and Renato Mancuso. The potential of programmable logic in the middle: cache bleaching. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 296–309. IEEE, 2020.

[176] RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification *Requirements and Technical Concepts for Aviation*, 1992.

[177] Svetlana Sagadeeva and Matthias Boehm. Sliceline: Fast, linear-algebra-based slice finding for ml model debugging. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 2290–2299, New York, NY, USA, 2021. Association for Computing Machinery.

[178] Ignacio Sañudo, Roberto Cavicchioli, Nicola Capodieci, Paolo Valente, and Marko Bertogna. A survey on shared disk i/o management in virtualized environments under real time constraints. *ACM SIGBED Review*, 15(1):57–63, 2018.

[179] Stuart E Schechter, Rachel A Greenstadt, and Michael D Smith. Trusted computing, peer-to-peer distribution, and the economics of pirated entertainment. In *Proceedings of The Second Annual Workshop on Economics and Information Security*, pages 29–30. Citeseer, 2003.

[180] Madeline Schiappa, Graham Chantry, and Ivan Garibay. Cyber security in a complex community: A social media analysis on common vulnerabilities and exposures. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 13–20. IEEE, 2019.

[181] Bernhard Schölkopf and Alex J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* . MIT PressCambridge, MA, 2002.

[182] Sergej Schumilo, Cornelius Aschermann, Ali Reza Abbasi, Simon Wörner, and Thorsten Holz. HYPER-CUBE: High-Dimensional Hypervisor Fuzzing. In *NDSS*, 2020.

[183] Sergej Schumilo, Cornelius Aschermann, Ali Reza Abbasi, Simon Wörner, and Thorsten Holz. Nyx: Greybox Hypervisor Fuzzing using Fast Snapshots and Affine Types. In *USENIX Security Symposium*, 2021.

[184] Hex Five Security. MultiZone Security for RISC-V. 2021.

[185] Alexander Senf, Robert Davies, Frédéric Haziza, John Marshall, Juan Ramón Troncoso-Pastoriza, Oliver Hofmann, and Thomas M. Keane. Crypt4gh: a file format standard enabling native access to encrypted data. *Bioinformatics*, 37(17):2753–2754, 2021.

[186] Yasser Shalabi, Mengjia Yan, Nima Honarmand, Ruby B. Lee, and Josep Torrellas. Record-Replay Architecture as a General Security Framework. *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 180–193, 2018.

[187] Ashleigh K Shelton and Paul Skalski. Blinded by the light: Illuminating the dark side of social network use through content analysis. *Computers in Human Behavior*, 33:339–348, 2014.

[188] Youren Shen, Hongliang Tian, Yu Chen, Kang Chen, Runji Wang, Yi Xu, Yubin Xia, and Shoumeng Yan. Occlum: Secure and efficient multitasking inside a single enclave of intel SGX. In James R. Larus, Luis Ceze, and Karin Strauss, editors, *ASPLOS '20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, March 16-20, 2020*, pages 955–970. ACM, 2020.

[189] Zekun Shen, Ritik Roongta, and Brendan Dolan-Gavitt. Drifuzz: Harvesting Bugs in Device Drivers from Golden Seeds. In *USENIX Security Symposium*, 2022.

[190] Siemens. *Infrastructure in the Cloud*. Last accessed:12/22.

[191] Siemens AG. Jailhouse hypervisor source code.

[192] Janice C Sipior, Burke T Ward, and Ruben A Mendoza. Online privacy concerns associated with cookies, flash cookies, and web beacons. *Journal of internet commerce*, 10(1):1–16, 2011.

[193] Parul Sohal, Rohan Tabish, Ulrich Drepper, and Renato Mancuso. Profile-driven memory bandwidth management for accelerators and cpus in qos-enabled platforms. *Real-Time Systems*, pages 1–40, 2022.

[194] Liubisa Stanković, Milos Daković, and Ervin Sejdić. Introduction to graph signal processing. *In Vertex-Frequency Analysis of Graph Signals*, 27(66):3–108, 2019.

[195] Udo Steinberg and Bernhard Kauer. NOVA: A microhypervisor-based secure virtualization architecture. In *Proceedings of the 5th European conference on Computer systems*, pages 209–222, 2010.

[196] G Edward Suh, Charles W O'Donnell, and Srinivas Devadas. Aegis: A single-chip secure processor. *IEEE Design & Test of Computers*, 24(6):570–580, 2007.

[197] Marco Antonio Tangaro, Giacinto Donvito, Marica Antonacci, Matteo Chiara, Pietro Mandreoli, Graziano Pesole, and Federico Zambelli. Laniakea: an open solution to provide Galaxy "on-demand" instances over heterogeneous cloud infrastructures. *GigaScience*, 9(4), April 2020.

[198] Thales. *TransVital*. Last accessed:12/22.

[199] Thales, SYSGO, Fraunhofer IESE, University of Rostock, ESE, and DB Netz. *Research Report - SIL4 Cloud*. Last accessed:12/22.

[200] Martin Thomson. A privacy analysis of google's topics proposal. 2023.

[201] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state-of-the-art, frameworks, trends, and challenges. *arXiv e-prints*, pages arXiv–2211, 2022.

[202] Chia-che Tsai, Donald E. Porter, and Mona Vij. Graphene-sgx: A practical library OS for unmodified applications on SGX. In Dilma Da Silva and Bryan Ford, editors, *2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017*, pages 645–658. USENIX Association, 2017.

[203] Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. *Proceedings of the International AAAI Conference on Web and Social Media*, 4(1):178–185, May 2010.

[204] Florian Turati. Analysing and exploiting google's floc advertising proposal. Master's thesis, ETH Zurich, Department of Computer Science, 2022.

[205] Anuj Vaishnav, Khoa Dang Pham, and Dirk Koch. A survey on FPGA virtualization. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 131–1317, 2018.

[206] Luca Vassio, Michele Garetto, Emilio Leonardi, and Carla Fabiana Chiasserini. Mining and modelling temporal dynamics of followers' engagement on online social networks. *Social Network Analysis and Mining*, 31:012012, 01 2022.

[207] Luca Vassio, Danilo Giordano, Martino Trevisan, Marco Mellia, and Ana Paula Couto da Silva. Users' fingerprinting techniques from tcp traffic. In *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pages 49–54, 2017.

[208] Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the international workshop on software fairness*, pages 1–7, 2018.

[209] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. Graviton: Trusted execution environments on gpus. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 681–696, USA, 2018. USENIX Association.

[210] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[211] Wei Wang, Miodrag Bolic, and Jonathan Parri. pvFPGA: paravirtualising an FPGA-based hardware accelerator towards general purpose computing. *International Journal of High Performance Computing and Networking*, 10(3):179–193, 2017.

[212] Wei Wang, Zhiyu Hao, and Lei Cui. ClusterRR: a record and replay framework for virtual machine cluster. *Proceedings of the 18th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2022.

[213] Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, Sathyanarayanan Manamohan, Saikat Mukherjee, Vishesh Garg, Ravi Sarveswara, Kristian Händler, Peter Pickkers, N Ahmad Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 594(7862):265–270, 2021.

[214] Andrew Waterman and Krste Asanović. The RISC-V Instruction Set Manual Volume I: Unprivileged ISA. 2019.

[215] Andrew Waterman and Krste Asanović. The RISC-V Instruction Set Manual Volume II: Privileged Architecture. 2019.

[216] Holger Wendland. *Scattered Data Approximation* , volume 17. Cambridge Univ. Press, 2005.

[217] Cornelia Wulf, Michael Willig, and Diana Göhringer. A survey on hypervisor-based virtualization of embedded reconfigurable systems. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pages 249–256. IEEE, 2021.

[218] Xen. *Xen Populate On Demand*. Last accessed:12/22.

[219] Tian Xia, Ye Tian, Jean-Christophe Prévotet, and Fabienne Nouvel. Kerone: A new hypervisor managing fpga reconfigurable accelerators. *Journal of Systems Architecture*, 98:453–467, 2019.

[220] Xilinx. Enabling Virtualization with Xen Hypervisor on Zynq UltraScale+ MPSoCs (White Paper WP474). Technical report, 2016.

[221] Qiuchen Yan and Stephen McCamant. Fast PokeEMU: Scaling Generated Instruction Tests Using Aggregation and State Chaining. *Proceedings of the 14th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2018.

[222] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[223] Xiao Yang, Seungbae Kim, and Yizhou Sun. How do influencers mention brands in social media? sponsorship prediction of instagram posts. In *International Conference on Advances in Social Networks Analysis and Mining*, New York, NY, USA, 2019. Association for Computing Machinery.

[224] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.

[225] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A fairness-aware incentive scheme for federated learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 393–399, 2020.

[226] Shuai Yuan, Jun Wang, and Xiaoxue Zhao. Real-time bidding for online advertising: measurement and analysis. In *Proceedings of the seventh international workshop on data mining for online advertising*, pages 1–8, 2013.

[227] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Steven D. Gribble and Dina Katabi, editors, *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*, pages 15–28. USENIX Association, 2012.

[228] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[229] Yuanhai Zhang, Hui Han, Jinxing Jiao, Guizhou Xu, Gang Chen, and Kai Huang. Brief industry paper: Sylixos: A secure and compatible rtos with constant scheduling on smp. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 449–452. IEEE, 2021.

[230] Mark Zhao, Mingyu Gao, and Christos Kozyrakis. Shef: Shielded enclaves for cloud fpgas. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 1070–1085, New York, NY, USA, 2022. Association for Computing Machinery.

[231] ETH Zurich. Ariane: An open-source 64-bit RISC-V Application Class Processor. 2021.