Universidade de Aveiro
2026

Tomás
Carvalho Bogalho

Plataforma de computação confidencial com SGX

Confidential computing platform with SGX

Universidade de Aveiro
2026

Tomás
Carvalho Bogalho

Plataforma de computação confidencial com SGX

Confidential computing platform with SGX

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Cibersegurança, realizada sob a orientação científica do Doutor André Ventura da Cruz Marnôto Zúquete, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, do Doutor Tomás António Mendes Oliveira e Silva, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

agradecimentos /
acknowledgements

Agradeço toda a ajuda a todos os meus colegas e companheiros.

palavras-chave

computação confidencial, Intel SGX, posicionamento WiFi, monitorização de ocupação, análise com preservação da privacidade, gestão segura da memória

resumo

A presença de infraestrutura de rede em ambientes académicos oferece uma oportunidade para extrair indicadores de ocupação, como ocupação de salas e frequência, no entanto o processamento de logs de ligação brutos apresentam riscos à privacidade.
A computação confidencial, particularmente Intel SGX, resolve isso ao permitir a execução segura dentro de enclaves protegidos por hardware, garantindo que os dados permaneçam inacessíveis pela infraestrutura subjacente.
No entanto, as implementações SGX padrão enfrentam dificuldades com o processamento de dados em grande escala devido ao tamanho limitado da cache da página do enclave e à sobrecarga de desempenho da paginação nativa.
Este trabalho propõe uma arquitetura de processamento seguro centralizada com um mecanismo de paginação de software personalizado no nível ao nível da aplicação, desenhado para superar essas limitações.

keywords confidential computing, Intel SGX, WiFi positioning, occupancy monitoring, privacy-preserving analytics, secure memory management

abstract

The presence of network infrastructure in academic environments provides an opportunity to extract occupancy indicators, such as room occupancy and attendance, yet processing raw connection logs introduces privacy risks.
Confidential Computing, particularly Intel SGX, addresses this by enabling secure execution within hardware-protected enclaves, ensuring data remains unaccessed by the underlying infrastructure.
However, standard SGX implementations struggle with large-scale data processing due to the restricted Enclave Page Cache size and the performance overhead of native paging.
This work proposes a centralized secure processing architecture featuring a custom, application-level software paging mechanism designed to overcome these limitations.

acknowledgement of use of AI tools

Recognition of the use of generative Artificial Intelligence technologies and tools, software and other support tools.

I acknowledge the use of Deepl (https://www.deepl.com/) for translation assistance and the use of Gemini Pro Model (Google Gemini, https://gemini.google.com) to summarize longer articles and to verify the consistency, grammar and articulation of phrases.

Contents

List of Figures

List of Tables

Acronyms

AP	Access Point
ASP	AMD Secure Processor
CPU	Central Processing Unit
CSI	Channel State Information
CVM	Confidential Virtual Machine
DCAP	Data Center Attestation Primitives
DMA	Direct Memory Access
DNN	Deep Neural Networks
DPR	Dynamic Partial Reconfiguration
DRAM	Dynamic Random-Access Memory
DRM	Digital Rights Management
EAP	Extensible Authentication Protocol
EDL	Enclave Definition Language
ELRANGE	Enclave Linear Address Range
EPC	Enclave Page Cache
EPCM	Enclave Page Cache Metadata
FPGA	Field-Programmable Gate Arrays
GPU	Graphics Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IoT	Internet of Things
MAC	Media Access Control
MKTME	Multi-key Total Memory Encryption
ML	Machine Learning
MMU	Memory Management Unit
NS	Non Secure
OS	Operating System
PoC	Proof of Concept
PRM	Processor Reserved Memory
PWR	Passive WiFi Radar
RD	Research and Development
REE	Rich Execution Environment
ROM	Read-Only Memory
RSSI	Received Signal Strength Indicator
SCR	Secure Configuration Register
SDK	Software Development Kit
SDM	Software Development Manual
SEAM	Secure-Arbitration Mode
SECS	SGX Enclave Control Structure
SEV	Secure Encrypted Virtualization
SEV-ES	Secure Encrypted Virtualization Encrypted State
SEV-SNP	Secure Encrypted Virtualization Secure Nested Paging
SGX	Software Guard eXtensions
SMC	Secure Monitor Call
SMI	System Management Interrupt
SMM	System Management Mode
SoC	System-on-Chip
SQL	Structured Query Language
SP	Service Provider
SRAM	Static Random-Access Memory
SVSM	Secure Virtual Machine Service Module
TA	Truste Application
TC	Trusted Computing
TCB	Trusted Computing Base
TD	Trusted Domain
TDX	Trust Domain Extensions
TEE	Trusted Execution Environment
TPM	Trusted Platform Module
TTP	Trusted Third Party
TZASC	TrustZone Address Space Controller
TZMA	TrustZone Memory Adapter
TZPC	TrustZone Protection Controller
UU	Universal User
VC3	Verifiable Confidential Cloud Computing
VM	Virtual Machine
VMPL	Virtual Machine Protection Level
VMX	Virtual Machine Extensions
VT	Virtualization Technology
XFRM	Extended Features Request Mask

CHAPTER 1
Introduction

The presence of wireless networks in everyday infrastructures has evolved beyond connectivity, due to their large presence and usage, it has become a reliable source of data. In a campus environment, the metadata generated [1] by user connections to the WiFi network, like the eduroam network, allows the somewhat precise localization of individuals, from here several indicators can be extracted such as room occupancy rates, the verification of class attendance, and the understanding of movement flows within the campus.

However, the extraction of statistical information from this raw data creates a significant difficulty between utility and privacy, namely the correlation between authentication logs and the physical location of an Access Point (AP) with class schedules and student lists exposes a significant quantity of sensitive information to the infrastructure. Current security models protect the data at rest and during transit with encryption and hashing, but often leave it vulnerable during computation, where data needs to be in cleartext, exposed in memory and to the Operating System (OS), in order to be processed.

To resolve this, a paradigm known as confidential computing occurs, unlike privacy-preserving techniques that rely on obfuscation or anonymization, confidential computing uses TEEs to protect data. This ensures that only code that is authorized and authenticated can process data, without exposing it to the underlying infrastructure or privileged users.

1.1 Motivations

The University of Aveiro possesses a large wireless infrastructure that generates connection logs whenever a user attempts to connect to an AP. Since this infrastructure is already in place, it represents an opportunity to extract statistical indicators without additional investment. However, this data remains unused due to the privacy risks associated with handling of the logs.

[1]Refers to data that describes the content, formats, or attributes of a data record or information resource, description by de Keyser [10]

Moreover, relying on implicit trust in the internal network is a vulnerability in itself, as modern security paradigms dictate, no infrastructure component should be considered permanently trustable. Under the same assumption, if the internal network or the privileged infrastructure were to be compromised, sensitive user data would be immediately exposed. Furthermore, processing this data on-premise limits scalability, by adopting a secure processing model with SGX, the University creates the possibility to offload these computational tasks without compromising data confidentiality or integrity.

However, while SGX is supported by the hardware infrastructure currently available, and offers the secure capabilities required to process sensitive information, both remote and locally, it presents a technical challenge, the processing of large volumes of data within an enclave. Since execution is restricted by the memory limits of the EPC, and standard implementations lack efficient native swapping, applications can quickly exhaust the available secure memory.

Therefore, the primary motivation of this work is to bridge the gap between the security of TEEs and the processing of large volume data, this requires the development of an architecture capable of managing memory efficiently.

1.2 Objectives

The main goal of this dissertation is to deliver a Proof of Concept (PoC) of an Intel SGX implementation that executes authorized applications to produce aggregated data from raw WiFi logs. This process aims to generate statistical data for the university without compromising the anonymity of WiFi users or revealing any sensitive information.

Specifically, the system aims to ingest network events, such as connection and disconnection timestamps and AP identifiers, and cross-reference them with institutional data. By correlating these logs with class schedules and student enrollment lists, which will also be used as input to the application, the system aims to extract occupancy metrics, such as the number of students per room and cumulative attendance records per student. Furthermore, the system will also aim to support configurable time windows, allowing these metrics to be calculated over varying intervals, ranging from the past week to an entire semester.

However, extracting and processing these indicators in a trusted environment presents technical challenges, like the limits of the secure memory available to an enclave and the lack of efficient swapping due to security reasons. The execution environment should then provide essential primitives for various programs, such as memory usage monitors, the secure ingestion of encrypted data from external sources, and the production of results for various destinations, these mechanisms should include an equivalent to a swap area to allow the processing of large datasets despite the internal enclave memory, or a snapshot capability to allow discontinuous processing.

In order to explore a solution to this problem, the current work focuses on the steps below:

1. Elaboration of the state-of-the-art regarding WiFi-based indoor positioning, the application of SGX in this domain, and techniques for large-scale data processing.
2. Definition of a generic architecture capable of processing large volumes of data, supporting simultaneous processing or over extended periods.

3. Development and documentation of a PoC application to demonstrate the proposed architecture.
4. Experimental validation of the system using raw WiFi logs to extract useful aggregated statistical indicators, verifying the accuracy of the results.

1.3 Outline

Chapter 2 provides the necessary context for the dissertation by introducing the concept of sensitive data and the risks associated with processing it. It also covers an overview of TEEs and Trusted Computing (TC) concepts, elaborating an analysis of technologies including ARM TrustZone, AMD SEV and Intel TDX. Furthermore, it details the architecture of Intel SGX, the core technology of this thesis, covering its lifecycle, architecture, components, security mechanisms and performance limitations.

Chapter 3 reviews the state-of-the-art in indoor positioning systems, focusing on methodologies utilizing WiFi infrastructure. It examines the privacy implications found in the literature, categorized from the omission of mechanisms to data minimization and opt-in models. Additionally, this chapter analyzes previous applications of SGX in location-based services and reviews strategies for processing large-scale data within enclaves.

CHAPTER 2

Context

The present chapter provides a comprehensive understanding of the technical underpinnings of SGX and the protection of sensitive data. It begins with the introduction of the concept of sensitive data, followed by TEEs, giving an overview of this type of mechanism. Then the chapter compares technologies, analyzing the distinctions between several TEEs implementations. After this, the discussion focuses on Intel SGX, detailing its architecture, workflow, security mechanisms and technical challenges associated with data processing within an enclave.

2.1 Sensitive Data

Driven by technological advances and institutional needs, nearly all buildings are now equipped with internet access provided through APs. These allow for the determination of user location with a reasonable degree of precision, as users must connect to a specific AP to access the network.

Although often categorized as technical metadata, it presents privacy challenges due to its potential to uncover sensitive information. While raw location logs may not possess intrinsic sensitiveness, like a direct medical diagnosis might, it is possible to acquire sensitive information through their processing context. As Quinn [11] states, the critical factor is the computational distance required to infer sensitive information from apparently neutral data.

In the case of indoor positioning, this computational distance is low, the aggregation of location traces creates a high risk of linkability, where disparate datasets, such as anonymous WiFi logs and public class schedules, can be easily cross referenced. This reduces the effort necessary to de-anonymize users, transforming technical data into identifiable information.

It is relevant to mention that the collection of connection logs is not driven by the production of occupancy analytics, but from a contractual obligation imposed on institutions, like the University of Aveiro, that implement the eduroam network. As stated in Global eduroam Governance Committee [12], a Service Provider (SP) is required to maintain logging information for a minimum retention period of three months, although this duration may vary depending on national regulations. Specifically, this logging includes the timestamps of authentication requests, the outer Extensible Authentication Protocol (EAP) identity, the client's Media Access Control (MAC) address, and the identification of the specific AP used for the connection, which will be used in the course of this dissertation.

The following list exemplifies sensitive information used for indoor location mentioned throughout this document:

- **MAC Address:** A persistent hardware identifier assigned to the network interface controller, this serves as a fingerprint that uniquely identifies a user's device.
- **Universal User (UU):** The unique user identifier, such as a student ID, employee username, or email, required for network authentication.
- **Connection Logs:** Time records containing precise association and disassociation timestamps, these allow for the reconstruction of a user's daily routine, duration of stay, and punctuality patterns.
- **Spatial Hierarchy:** Topological data classifying location from macro to micro levels, like the location of the AP the user connected to.
- **Visual and Biometric Data:** In systems utilizing optical sensors, raw data includes high-resolution images or video capable of revealing biometric features and behavioral signatures.

2.2 Trusted Execution Environment (TEE)

Before defining TEEs, it is essential to understand the underlying concept of TC, as defined by Sabt *et al.* [1], a TC aims to enable systems to achieve secure computation, privacy, and data protection. Historically, TC relied on a dedicated hardware component to provide a functional interface for security operations, one common example is the Trusted Platform Module (TPM), which allows a system to prove its integrity and securely store cryptographic keys, however, a limitation of the TPM is its inability to perform isolated execution of arbitrary code, since it is mainly designed for passive storage and reporting, rather than active processing. But more recent approaches to TC, shift the execution of arbitrary code to a trusted environment that allows tamper resistant execution of its applications.

While various terminologies exist in the literature, this dissertation adopts the term TEE, this is defined by Sabt *et al.* [1] as a secure, integrity protected computational environment, typically with isolated memory and specific storage capabilities. Using a combination of hardware and software components in the Central Processing Unit (CPU), it isolates specific workloads, commonly referred to as Truste Application (TA), from the rest of the system. Its primary objectives are to guarantee the authenticity of the executed code, the integrity of the runtime state and the confidentiality of the data, like CPU registers, while protecting code, memory, and sensitive Input/Output (I/O) from unauthorized access, in addition a TEE should have the ability to support remote attestation, allowing third parties to verify the trustworthiness of the environment, although specific implementation details may vary across different types of TEEs.

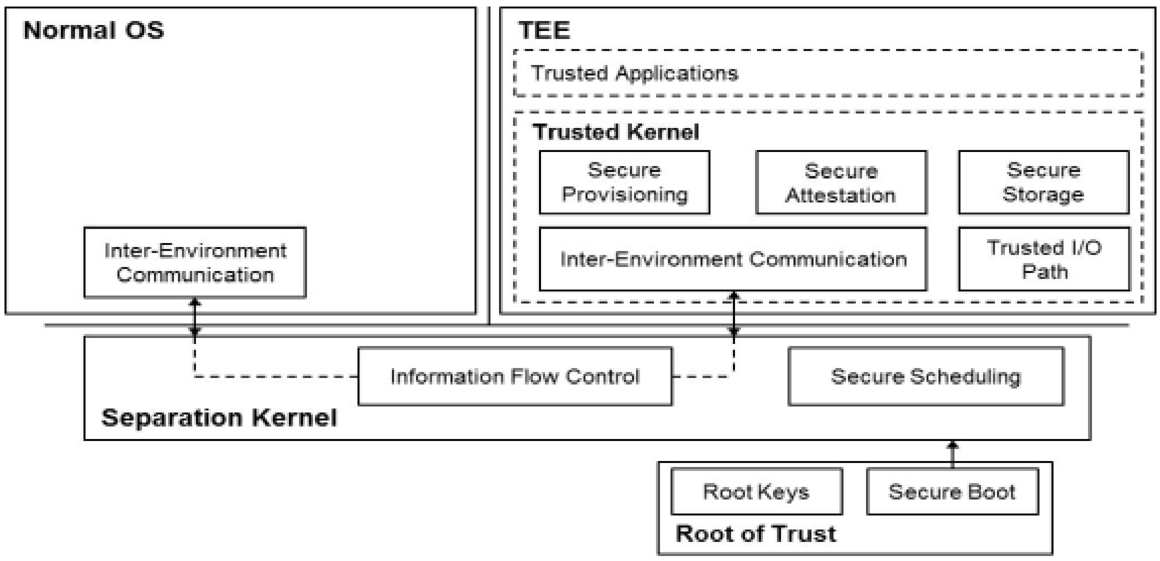

Figure 2.1: Overview of TEE building blocks, image by Sabt *et al.* [1].

The implementation of these can differ, leading to a categorization of TEEs based on their enforcement mechanism, hardware-based TEEs rely on specific hardware extensions, such as Intel SGX, to enforce isolation at the lower level, effectively anchoring trust in the physical processor, in contrast, software-based TEEs depend on a software layer typically a hypervisor or microkernel to protect the execution environment.

As seen in the Figure 2.1 by Sabt *et al.* [1], a TEE in composed by building blocks such as:

- Secure Boot: guarantees that only code of a certain property can be loaded during boot.
- Secure Scheduling: provides a balanced coordination between the TEE and the rest of the system.
- Inter-Environment Communication: Interface that allows the TEE to communicate with the rest of the system.
- Secure Storage: Storage where confidentiality, integrity and authorized access of stored data is guaranteed.
- Trusted I/O Path: Communication between I/O peripherals with authenticity and sometimes confidentiality.

In contrast to the isolation of the TEE, the remaining portion of the system is often referred to as the REE or the untrusted environment, this includes the traditional operating system, the hypervisor, and the standard application stack (yellow component of Figure 2.2). While the REE is optimized for performance, versatility, and accessing peripherals, it loses the security guarantees required for sensitive data processing, as it makes it vulnerable to software attacks.

The architectural principle of trusted computing is the strict separation between these two worlds, the TEE protects its assets even when the surrounding REE is fully compromised, this separation can be observed in the Figure 2.2.

A TEE technology provides protection for various applications by enforcing isolation and integrity, as mentioned in the systematic review by Paju *et al.* [13], the scenarios for this

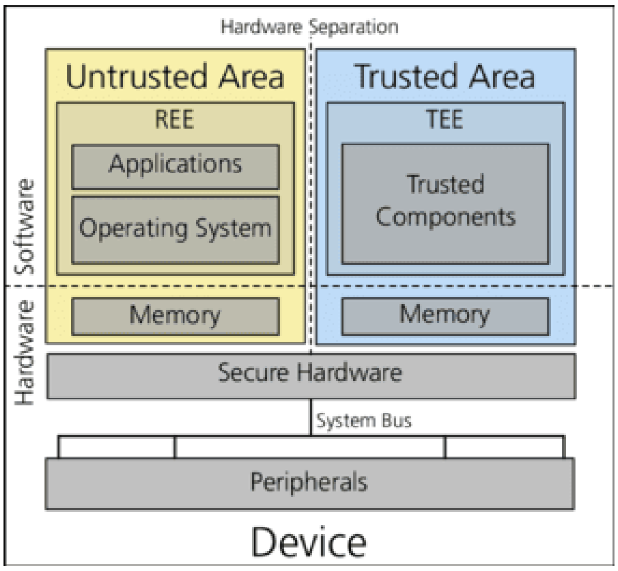

Figure 2.2: Overview of REE and of TEE, image by González [2].

technology appear in multiple domains, ranging from consumer electronics to critical cloud infrastructure.

In digital media, rights holders frequently employ TEEs for Digital Rights Management (DRM) to prevent the unauthorized copying of video or audio content, by establishing a hardware channel between the TEE and the device's display, the system blocks the device owner or malicious software from intercepting decoded media.

Financial applications also benefit from this technology, TEEs can be used to secure mobile wallets, peer-to-peer payments, and point-of-sale terminals. Furthermore, in blockchain ecosystems, they enable privacy for lightweight clients without compromising the performance of full nodes, acting as trusted backend systems to facilitate secure transactions.

In authentication TEEs are used to protect biometric identity methods such as facial recognition and fingerprint sensors. For example, the Android OS stores fingerprint data within the TEE, making it inaccessible to the OS environment. This isolation capability is also utilized to support passwordless standards, such as Apple's passkeys, by combining biometric verification with securely stored cryptographic private keys.

In the context of cloud computing and data analysis, areas relevant to this dissertation, TEEs provide a critical layer of defense. In a trusted cloud model, they protect the infrastructure against compromise, even if the backend or host OS is malicious, the adversary remains unable to access the specific TA running inside the enclave. This is also common for Machine Learning (ML) in sensitive fields like healthcare, allowing computation on confidential datasets without exposing the underlying information.

Finally, TEEs also contribute to system stability through runtime integrity and secure modular programming. They can enforce real-time kernel protection by isolating security monitoring services, potentially shutting down systems if kernel binaries are targeted. Additionally, they encourage secure software architectures by decoupling functionalities into self-contained modules, where the execution of one module is protected from the vulnerabilities of others.

2.3 TEE Technologies

Several TEE implementations exist, such as ARM TrustZone, AMD SEV, and Intel's most recent TDX. This chapter provides a comparative overview of these technologies, however, Intel SGX, the focus of this dissertation, is reviewed in greater detail in the subsequent section. Intel TDX is examined thereafter, as it represents the natural evolution of the SGX architecture.

2.3.1 ARM TrustZone

Initially introduced in 2004, ARM TrustZone is a set of hardware security extensions available on ARM processors, while first implemented on the Cortex-A series, it was later adapted for the more recent Cortex-M series, with some architectural differences suited for microcontrollers. Contrary to a standalone component, TrustZone is an approach designed to be integrated into a System-on-Chip (SoC), securing not just the processor but the entire electronic device, as described by Pinto and Santos [3], in this model the architecture is centered around the concept of distinct domains, the Secure World, the trusted environment, and the Normal World, the untrusted environment, as seen in Figure 2.3.

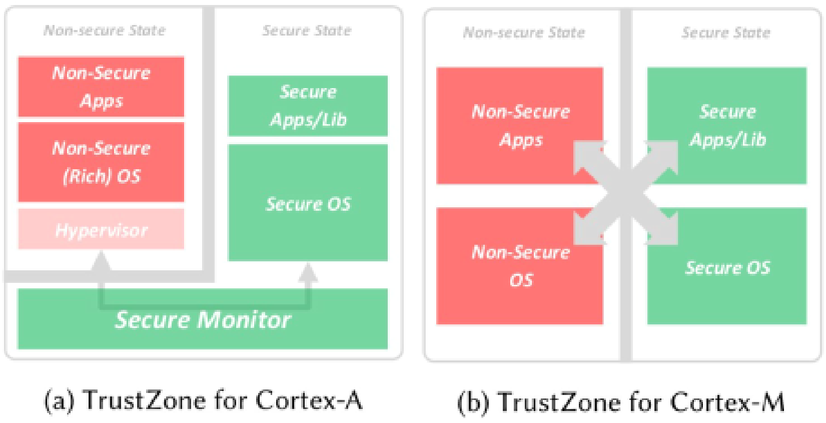

Figure 2.3: ARM TrustZone technology, image by Pinto and Santos [3].

This isolation allows software to execute with different privilege levels, preventing untrusted applications in the Normal World from directly accessing resources allocated to the Secure World. The state of the processor is determined by the Non Secure (NS) bit, typically located in the Secure Configuration Register (SCR), which effectively partitions the system's hardware resources.

To manage the transition between worlds, TrustZone employs a monitor mode, which executes in a secure state independent of the NS bit, this mode bridges the two software stacks and is entered via the privileged Secure Monitor Call (SMC) instruction or through specifically configured hardware exceptions.

Hardware isolation is reinforced by banked registers, which guarantees that security critical system registers are either hidden from the untrusted environment or are strictly accessed from the trusted environment. Regarding memory infrastructure, TrustZone introduces components such as the TrustZone Address Space Controller (TZASC) for Dynamic Random-Access Memory (DRAM) and the TrustZone Memory Adapter (TZMA) for off-chip Read-Only Memory (ROM)/Static Random-Access Memory (SRAM), these controllers partition memory into trusted and untrusted regions, preventing the Normal World from accessing secure memory while allowing the Secure World full access. TrustZone Memory Management Unit (MMU) maintains separate virtual to physical translation tables for each domain, and cache isolation is enforced via an additional tag bit on cache lines. Finally, the TrustZone Protection Controller (TZPC) extends this isolation to system peripherals, restricting access to specific devices based on the security domain.

In recent years, academic and industrial interest in this technology has surged, driven by the availability of technical documentation and accessible development platforms from manufacturers like Xilinx, this transparency has facilitated adoption within the community, particularly in the context of Research and Development (RD) in Internet of Things (IoT) devices.

While TrustZone is presented the standard for TEEs in the embedded ecosystem, its adoption in this work is constrained by the architectural model of the proposed solution. The current approach focuses on the centralized processing of aggregated connection logs exported by the network infrastructure, rather than computing directly on the edge devices. Deploying custom trusted applications directly onto the university's APs is impractical, as it would require proprietary access to the firmware and administrative privileges, or the acquisition of specific hardware that would lead to additional costs.

2.3.2 AMD SEV

Introduced in 2016, SEV is stated to be the first x86 technology that was designed to isolate a Virtual Machine (VM) from the hypervisor, since historically these have been considered trusted components, this allows, for example, clients in the cloud to secure their VM workload from the cloud administrator, keeping their data confidential and minimizing exposure. AMD did this using main memory encryption in SEV, with this, individuals VMs were assigned an individual key to encrypt all data, preventing the hypervisor from reading clear text

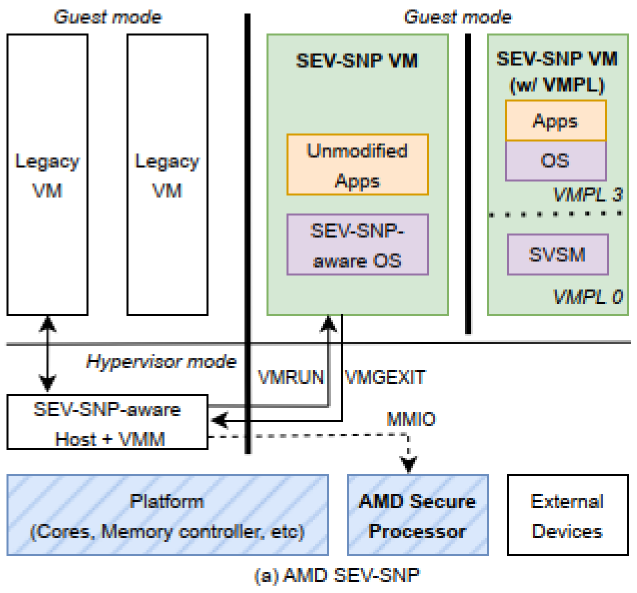

Figure 2.4: AMD SEV Architecture, image by Misono *et al.* [4].

information, as cited by Kaplan [14]. Later in 2017, a new version came out, the AMD Secure Encrypted Virtualization Encrypted State (SEV-ES) that added the additional feature of CPU register state protection, in order to do this, in SEV-ES the VM register state is encrypted at each hypervisor transition, preventing the hypervisor from seeing the data that is being used, adding confidentiality to in-memory data.

The most recent version is the Secure Encrypted Virtualization Secure Nested Paging (SEV-SNP), built on top of the previous versions while adding hardware based protections like memory integrity protection to prevent attacks like data replay, memory re-mapping, it also implements protection in interrupt behavior.

This is, in the reviewed literature, according to Misono *et al.* [4], the first Confidential Virtual Machine (CVM), which are VMs designed to alleviate the programmability and usability challenges of the previously proposed trusted computing technologies, by providing abstraction at the level of the VM. This technology uses a dedicated processor, the AMD Secure Processor (ASP) to manage the security features, this is an ARM-based processor that is separated from the main 32-bit core, but directly integrated in the CPU, which also allows for remote attestation.

The Figure 2.4 shows the overview of the AMD SEV-SNP, where green zones ilustrate CVMs with unmodified software (orange) and with modified software (purple) and the blue areas are trusted components, the vertical black line indicates that each CVM is isolated from the host and other encrypted VMs. This VM executes in an isolated environment where its memory and state registers are encrypted via a unique key managed by the ASP, this safeguards the integrity of guest address translations and strictly isolates the execution by blocking System Management Interrupt (SMI) events and preventing access from the System Management Mode (SMM).

This architecture also introduces Virtual Machine Protection Level (VMPL), a four-tiered privilege hierarchy to traditional x86 rings that enables a Secure Virtual Machine Service Module (SVSM) running at the highest level VMPL to securely offer services like virtual TPM and live migration.

This approach was not selected for this implementation due to architectural and practical constraints, AMD SEV operates at the VM level, encrypting the entire guest memory, even though this mitigates the memory capacity limitations inherent to SGX, it significantly expands the Trusted Computing Base (TCB) to include the entire guest OS, increasing the attack surface. In contrast, SGX offers application-level isolation, excluding the OS from the trust boundary, which is more aligned with the goal of developing a lightweight solution.

2.4 Intel Software Guard Extensions (SGX)

In similarity with previous TEE implementations, SGX is a set of extensions for the Intel architecture with the goal to allow for integrity and confidentiality during computation in a computer where all privileged components might be compromised, as defined by Costan and Devadas [5].

This emerged as a solution to a recurring problem known as secure remote computation, which refers to the difficulty of executing software on a remote computer owned and maintained by an untrusted party. In this context, SGX is one of many attempts to address this challenge by leveraging trusted hardware. To achieve this, the hardware establishes a secure container and a service that enables secure computation for the user.

In simpler terms, this model relies on software attestation, a capability that allows the hardware to prove to a user that they are communicating with the intended application on non-compromised hardware and that the code has not been altered.

This is achieved through a cryptographic signature that certifies the hash of the enclave's content, the user verifies this signature against a root of trust in the manufacturer's hardware, confirming the validity of the attestation key.

2.4.1 Enclave

An important concept in SGX terminology is an enclave, as described by Cheng *et al.* [7], is a protected region within an application's address space that allows the isolation of sensitive code and data from the rest of the system.

To maintain the security properties and persistent identity of each enclave instance, SGX employs a control structure known as the SGX Enclave Control Structure (SECS). This structure contains the stored metadata for each individual enclave and is kept in a dedicated EPC page with the type `PT_SECS`. During the enclave's lifecycle, the SECS is the first page allocated and the last to be deallocated, serving as the persistent identifier, consequently, the system software utilizes the virtual address of the SECS to reference the specific enclave when issuing SGX instructions. Despite being mapped for identification purposes, the SECS page remains inaccessible to the trusted enclave code and the untrusted system software, due to the presence of sensitive information regarding the enclave.

With the enclave's identity and attributes anchored in the SECS, its execution context is defined by a window in the virtual address space, known as the Enclave Linear Address Range (ELRANGE). This range is used to map all code and sensitive data to the corresponding EPC pages and any memory access attempted by the enclave outside this defined area is automatically mapped to access non-EPC memory via the same virtual addresses, allowing for interaction with the rest of the application.

While the ELRANGE defines the memory boundaries of the enclave, the nature of its execution environment is defined by the `ATTRIBUTES` field within the SECS, via flags such as `DEBUG`, Extended Features Request Mask (XFRM), and `MODE64BIT`. The `DEBUG` is intended to be used during development, since when set, it enables debugging features that allow the reading and modification of the enclave's memory. The XFRM enforces the enclave code to run with the architectural extensions enabled by the compiler, done by defining the value for the `XCR0` register during execution. Finally, the `MODE64BIT` flag determines whether the enclave operates using the 64-bit architecture, likely for backward compatibility.

Once these operational parameters are defined in the metadata, the enclave must pass through a specific sequence of states to become executable. This process begins with creation, where the `ECREATE` instruction transforms a free EPC page into the enclave's SECS, initializing it with architectural parameters and setting the texttttINIT attribute to false, while in this state, the lifecycle enters the loading phase, where the system software uses `EADD` to allocate pages for code and data, while `EEXTEND` updates the enclave's cryptographic measurement. After loading, initialization is started by the `EINIT` instruction, that validates a token and sets the `INIT` attribute to true, sealing the enclave from modification and enabling execution in Ring 3. Finally, the destruction of the enclave occurs with the `EREMOVE` instruction, which deallocates pages by clearing their `VALID` bit, ensuring that the SECS page is only destroyed once all other pages associated with the enclave have been successfully removed, as shown in Figure 2.5.

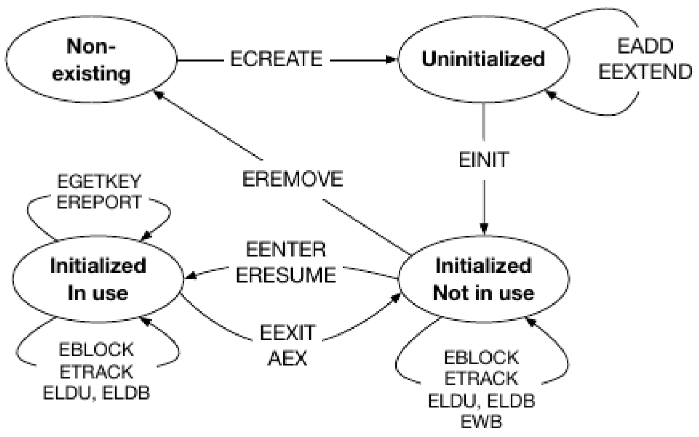

Figure 2.5: An Enclave Lifecycle, by Costan and Devadas [5].

2.4.2 Architecture and Workflow

To secure its memory from the untrusted environment, SGX reserves a region in DRAM known as the Processor Reserved Memory (PRM). This region is isolated by the CPU, blocking all non-enclave memory accesses from softwares such as the kernel or hypervisor, as well as protecting against SMM and Direct Memory Access (DMA) attacks from peripherals.

The PRM also contains the EPC, which consists of 4KB memory pages that store the enclave's code and data. While the allocation of these EPC pages is managed by the untrusted system software, the security is enforced by hardware. The CPU tracks the state and permissions of every EPC page in the Enclave Page Cache Metadata (EPCM) to guarantee that each page belongs to a specific enclave and cannot be mapped by unauthorized entities.

The workflow begins with the loading of code and data into the enclave, during this operation the untrusted system software asks the CPU to copy data from unprotected memory into the EPC pages and assigns them to the enclave. After the loading is completed, the system software indicates to the CPU to mark the enclave as initialized. At this point the loaded code becomes executable in the enclave and the loading method is disabled to prevent modifications.

The execution flow of an enclave is governed by specific CPU instructions, these are similar to context switching between user mode and kernel mode, but the enclave execution remains within Ring 3.

2.4.3 Enclave Page Cache

As mentioned previously, the content of the enclave and necessary data are data structures in the EPC as shown in the Figure 2.6.

The SGX design allows for multiple enclaves running at the same time, which translates to the necessity of a multi-process environment. This is possible due to the capability of the EPC to assign different pages to different enclaves. Non-trusted software cannot directly access it, since it is contained in the PRM.

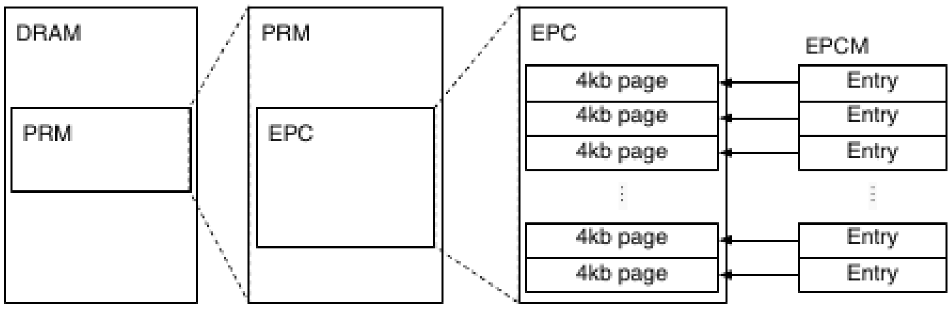

Figure 2.6: EPC Memory overview, by Costan and Devadas [5].

This mechanism is managed by the same system software that manages the rest of the computer's physical memory, which could be a hypervisor or the OS kernel. Both accomplish this with the use of SGX instructions to allocate and free pages in the enclaves. But as mentioned previously, this is not a trusted component, so the SGX processor verifies every allocation decision, and has the possibility to refuse them if they compromise any security guarantee.

To perform these verifications, SGX keeps some information about previous allocation decisions for each EPC page in the EPCM, an array with one entry per page. Three relevant fields in each entry that affect the ownership of the EPC page are `VALID`, a bit indicating whether an EPC page is currently active and allocated to an enclave, if set to one it signifies the page is in use, preventing reallocation or overwriting by other instructions, `PT`, eight bits that define the type of page and `ENCLAVESECS` which identifies the enclave that owns the page.

The contents of this map can only be read by an enclave as stated in Intel Software Development Manual (SDM).

2.4.4 ECALLs and OCALLs

In Intel SGX, `ECALL` and `OCALL` are mechanisms for communication between the untrusted application and the trusted enclave.

`ECALLs` allow the untrusted application to invoke a pre-defined function inside the enclave, passing input parameters and pointers to shared memory within the application.

On the other hand, `OCALLs` enable the enclave to invoke a pre-defined function in the untrusted application, but unlike `ECALLs`, cannot share enclave memory directly and must copy parameters into the application's memory before the call.

Both `ECALLs` and `OCALLs` are defined using the Enclave Definition Language (EDL), where `ECALLs` are declared in the trusted section and `OCALLs` in the untrusted section of the file. During the build process, the Edger8r [1] tool parses the EDL file and generates function wrappers around the actual functions.

[1]Tool provided by the Intel SGX Software Development Kit (SDK) that automatically generates the code that connects the untrusted application and the trusted enclave

2.4.5 Attestation, Sealing and Unsealing

SGX relies on software attestation to prove to a remote party that it is communicating with a specific piece of software running in a secure container hosted by trusted hardware. One of the most important components to this process is the enclave's identity, which is cryptographically established during its initialization and loading phases, as mentioned previously.

The identity of an enclave is primarily defined by two measurement registers located in the enclave's SECS. The first, `MRENCLAVE`, contains a hash that uniquely identifies the enclave's code and data contents, acting as a cryptographic log of the creation process, it incorporates every instruction used to build the enclave, consequently, any modification to the loaded code, data, or their order results in a completely different `MRENCLAVE` value, ensuring that the running software corresponds exactly to the developer's intent.

Complementing this is `MRSIGNER`, which identifies the entity that signed the enclave rather than the code itself. This register stores the hash of the public key modulus used to sign the `SIGSTRUCT` certificate, allowing remote parties to trust an enclave based on the author's reputation. The figure 2.7 shows how an enclave identity is established.

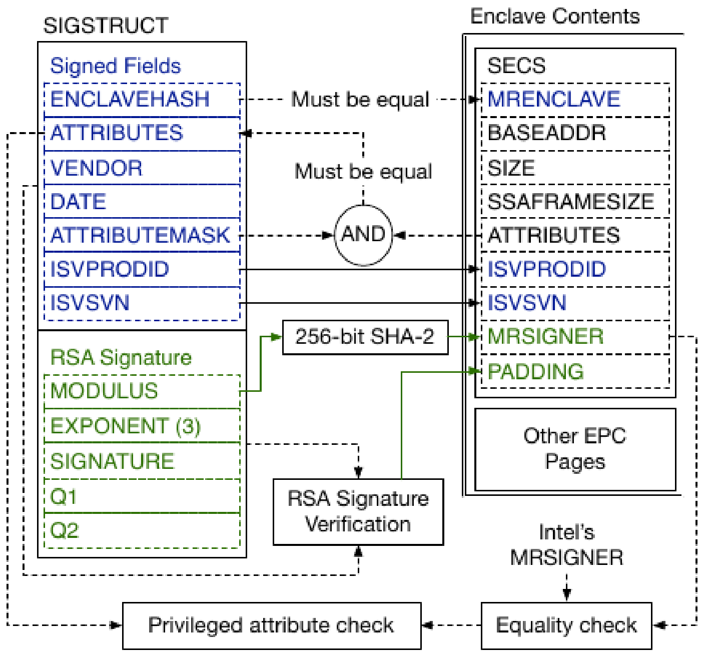

Figure 2.7: Establishing an Enclave's Identity, by Costan and Devadas [5].

The attestation process leverages these identities through a mechanism known as local attestation, where an enclave proves its identity to another enclave on the same platform using the `EREPORT` instruction. To extend this trust remotely, SGX uses a privileged enclave known as the Quoting Enclave, that receives the local report, verifies it, and replaces the local MAC with a cryptographic signature produced by the processor's unique attestation key. This signature, which includes the `MRENCLAVE` and `MRSIGNER`, serves as the final proof to the remote service provider that the enclave is authentic and running on genuine hardware.

The initialization process is finalized by the `EINIT` instruction, which validates the enclave's `SIGSTRUCT`, this checks that the measurement in the certificate matches the calculated `MRENCLAVE` and populates the `MRSIGNER` field in the SECS, defining the enclave's identity before it can begin execution.

2.5 Memory and Performance Limitations in SGX

Even though SGX provides confidentiality and integrity, it introduces performance overheads and resource constraints.

One constraint is the limited capacity of the EPC, since it resides within the PRM, where the memory is limited. When an application's working set exceeds the available EPC capacity, the system must perform paging operations to evict EPC pages to unprotected DRAM.

Unlike normal paging, SGX paging requires cryptographic operations, where the hardware encrypts the page before writing it to untrusted memory, and verify and decrypt it upon restoration. This causes a performance overhead, often referred to as the EPC Bottleneck, making large-memory workloads impractical without optimization. Various approaches appear

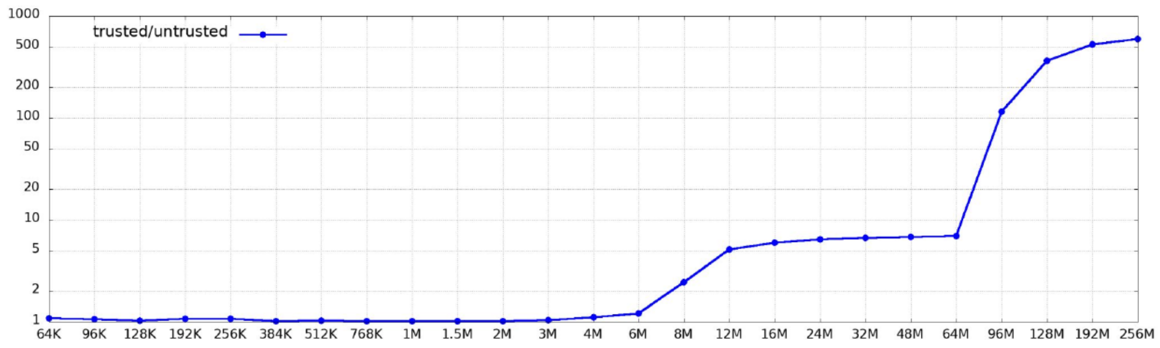

Figure 2.8: Performance Overhead of an Enclaves as a Function of Data Size, by Silva and Zúquete [6].

in related work, such as data partitioning and demand paging optimization, attempt to mitigate this by minimizing the frequency of page swaps.

In addition to memory capacity, the cost of entering (`ECALL`) and exiting (`OCALL`) an enclave is worth mentioning, since a domain transition involves several hardware operations, including flushes, security checks, and context saving, as shown in the figure 2.8.

This happens since SGX performance is dependent on the working set size, for data sizes under 64MB, the overhead remains manageable, however, a performance bottleneck is observed beyond 96MB.

This threshold corresponds to the physical limit of the EPC, once the enclave's memory usage exceeds this capacity, the OS is forced to perform secure paging, to swap them to untrusted memory. This results in a performance degradation, with execution times reaching up to 500 times that of native code due to the use of more clock cycles, showing the need for optimized memory management.

2.5.1 Intel TDX

Representing a natural evolution of the SGX architecture, Intel TDX expands the concept of hardware-enforced isolation from individual application enclaves to entire virtual environments, introduced with the 4th Generation Intel Xeon Scalable processors. This technology facilitates the deployment of confidential VMs, referred to as Trusted Domain (TD)s, under the supervision of the Secure-Arbitration Mode (SEAM), that provides encrypted CPU state and memory, with integrity protection and remote attestation, addressing scenarios where the host platform is considered untrustworthy within the threat model [2].

While TDX shares the same high-level goals as AMD SEV, it is built upon the trusted execution principles established by SGX. Internally, it uses similar architectural concepts, including the use of `ECALLs` transitions and the Data Center Attestation Primitives (DCAP) framework for remote attestation.

Technically, TDX combines these SGX foundations with virtualization technologies from other Intel technologies such as Virtualization Technology (VT) and Multi-key Total Memory

[2]Structured approach to identifying potential security risks, defining attacks from the perspective of an adversary to strengthen system defenses [15].

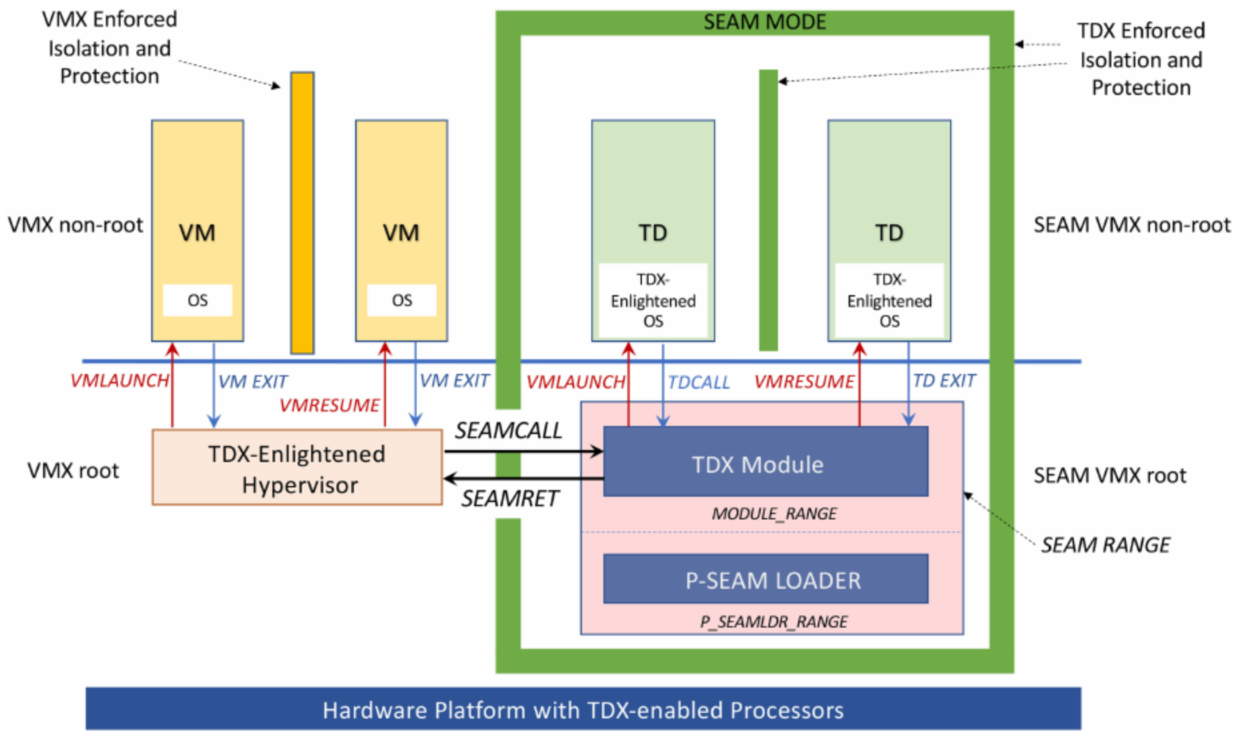

Figure 2.9: Intel TDX Architecture, image by Cheng *et al.* [7].

Encryption (MKTME), together, these components enforce the confidentiality and integrity of TD memory and CPU, ensuring that they can neither be accessed nor tampered with by other entities in the environment, such as the hypervisor or neighboring VMs in the same cloud tenant.

According to Cheng *et al.* [7] it achieves the previous using a combination of three components, namely, memory access control, runtime memory encryption and an Intel-signed TDX module that does the handling of security-sensitive TD management operations.

The Figure 2.9 illustrates the runtime architecture of the TDX technology. The two main components are the TDX compatible processors, that provide functional capabilities like hardware-assisted virtualization, memory encryption, integrity protection, and the TDX Module, that facilitates the manipulation of TD while applying security guarantees, besides this, the module provides two interface functions, a host-side interface and a guest-side interface. The previous module is loaded in to the SEAM range, a portion of memory that is reserved during boot. Another component present in this range is the SEAM loader, which is responsible for installing and updating the module. SEAM provides two execution modes, the SEAM Virtual Machine Extensions (VMX) root mode and non-root mode, a TDX compatible hypervisor operates in the VMX root mode and uses the `SEAMCALL` introduction to call host-side functions, functions that start with "TDH", of the module.

This is possible since the the logical processor changes VMX root mode into SEAM VMX root mode and starts executing code within the TDX Module, upon completing it returns in VMX root mode with the instruction `SEAMRET`. The TDs run in SEAM VMX non-root mode, since TDX supports the execution of unmodified user-level applications within a TD, like a normal VM, but the guest OS kernel (TDX-enlightened OS in Figure 2.9), needs to be altered in order to align with the TDX platform, namely managing TDX exceptions via an internal handler, implementation of a mechanism for communication between the TD and the TDX module, transitioning memory pages from private to shared I/O operations, although implementation details may vary according to the OS in use.

Although this technology effectively tackles the memory limitations imposed by SGX, eliminating the need for manual memory management in large data processing, its adoption in this work is constrained by current hardware availability. According to Intel specifications [3] , TDX requires 4th Generation Intel Xeon Scalable Processors, or newer, alongside specific platform support. Given that the computational infrastructure available for the development of this dissertation relies on earlier hardware generations that do not possess these extensions, the focus remains on SGX.

[3]https://www.intel.com/content/www/us/en/support/articles/000091103/processors/intel-xeon-processors.html

CHAPTER 3

Related Work

This chapter establishes the context for the proposed work by reviewing the related work in WiFi based indoor positioning and occupancy detection. Besides this, it examines the application of SGX in location based systems and analyzes strategies for large data processing within enclaves. While nearly identical solutions, combining already existing infrastructure monitoring with hardware enforced privacy, are scarce in the literature, distinct bodies of research provide the necessary basis for this dissertation.

The review begins by analyzing technical methodologies for indoor positioning using standard WiFi technologies, it then transitions to the critical challenge of user privacy in location analytics.

The following chapter analyzes SGX as a possibility for secure indoor positioning, evaluating its current status and performance in privacy preserving location based applications. Finishing with the specific analysis of data processing within SGX, reviewing architectural strategies for handling large scale sensitive data under enclave constraints.

3.1 Indoor Positioning System

As defined by Li *et al.* [8] an Indoor Positioning System is viewed as the grouping of three components: positioning principles and corresponding algorithms, technologies and hardware equipment. These are considered to have a meaningful impact on the performance of the positioning system (see Figure 3.1).

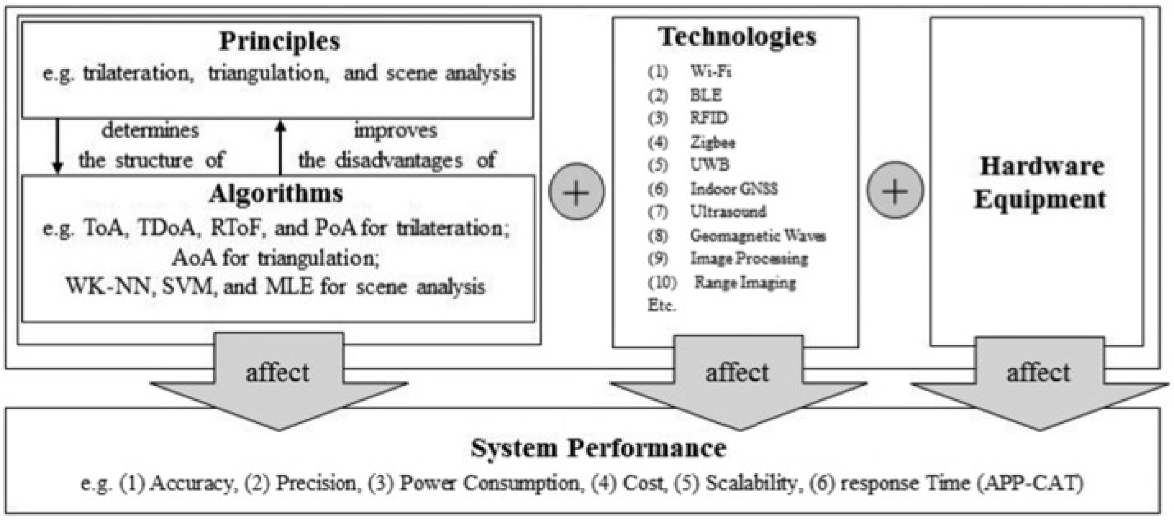
Principles
e.g. trilateration, triangulation, and scene analysis
determines the structure of
improves the disadvantages of
Algorithms
e.g. ToA, TDoA, RToF, and PoA for trilateration; AoA for triangulation; WK-NN, SVM, and MLE for scene analysis
Technologies
(1) Wi-Fi
(2) BLE
(3) RFID
(4) Zigbee
(5) UWB
(6) Indoor GNSS
(7) Ultrasound
(8) Geomagnetic Waves
(9) Image Processing
(10) Range Imaging
Etc.
Hardware Equipment
affect
affect
affect
System Performance
e.g. (1) Accuracy, (2) Precision, (3) Power Consumption, (4) Cost, (5) Scalability, (6) response Time (APP-CAT)

Figure 3.1: General structure of the model of an indoor positioning system by Li *et al.* [8].

3.1.1 Technologies used for Indoor Positioning System

There are several technologies used for an Indoor Positioning System, Li *et al.* [8] divided them in three categories, this overview has been expanded to include Optical and Environmental Sensors, reflecting the methodologies found in research literature, as shown by Table 3.1.

Table 3.1: Indoor Positioning Categories

Category	**Technologies**
Radio Frequency (RF)	WiFi
	Bluetooth Low Energy (BLE)
	Zigbee
	Ultra-wide band (UWB)
	Radio frequency identification (RFID)
	Indoor Global Navigation Satellite System (GNSS)
	Frequency modulation radio (FM-radio)
Non-Electromagnetic Waves	Ultrasound
	Geomagnetic Waves
Full-spectrum light	Range imaging
	Visible light positioning
	Laser
Optical	Cameras / Computer Vision
	Infrared
Environmental Sensors	CO2 Sensors

3.2 Wifi for Indoor Positioning

In the research literature, several terms related to indoor positioning appear interchangeably, such as positioning, localization, detection, counting, and tracking. While authors like Zou *et al.* [16] often distinguish these terms based on technical granularity, where localization implies pinpointing specific (x, y) coordinates and tracking involves temporal notion, this thesis adopts a broader operational perspective.

Instead of focusing on the precise coordinate level tracking of targets, the current work prioritizes occupancy detection and crowd estimation in a certain area, in this context, the specific terminology is less about the geometric precision of a single user's location and more about the aggregate presence of devices within a defined zone, treating location as a categorical state.

To achieve these various levels of granularity, ranging from simple presence detection to precise tracking, researchers leverage different characteristics of the WiFi signal. Consequently, the literature presents several distinct methodologies for indoor positioning, each utilizing a specific data metric. Prominent examples include the analysis of WiFi probe requests as demonstrated by Wang *et al.* [17], the measurement of Received Signal Strength Indicator (RSSI) performed by Zou *et al.* [16], and the more complex Channel State Information (CSI) techniques researched by Sobron *et al.* [18]. These approaches, along with others, form the technical basis for the studies reviewed throughout this chapter.

The following chapters will be divided according to the main goal of the Wifi for Indoor Positioning.

3.2.1 Energy Savings and Efficiency

Estimating building occupancy is a necessary step for optimizing building operations, in this scenario WiFi connection counts are presented as a cost efficient proxy for occupancy when comparing to most common methods, even showing a strong correlation with actual people counts as shown by Mohottige *et al.* [19].

In 2021, Alishahi *et al.* [20] proposed a framework that uses ML alongside with statistical analysis to extract occupancy indicators from WiFi connection counts, the authors argued that traditional sensors often suffer from latency, high maintenance costs, and limited scalability. In contrast, stated that associated WiFi device counts, which are devices attempting to authenticate with the AP not necessarily being able to, as a cost effective proxy for human presence. The framework showed positive results, however, Alishahi *et al.* [20] also highlighted limitations that impact the design of this thesis's methodology. It was noted that raw connection counts can possess a degree of noise and require calibration with ground-truth data to account for two sources of error, the stationary devices like printers, desktops, and projectors that permanently connect to the network must be filtered out and the device to occupant ratio, which is the variance in the number of devices carried by each user. The study emphasizes that these errors become more pronounced at smaller spaces, where attributing a connection to a specific zone is difficult without the precise localization data provided by metrics like RSSI.

In the 2020 Institute of Electrical and Electronics Engineers (IEEE) Radar Conference, Tang *et al.* [21] presented a methodology for people counting based on Passive WiFi Radar (PWR). The authors presented this approach to overcome the limitations of standard WiFi sensing, since RSSI methods are prone to unpredictable fluctuations and false positives due to multipath effects, and CSI techniques typically require specific hardware modifications or high-rate transmissions that degrade network throughput. The overall results presented a high accuracy of 99.54% for tasks of occupancy detection and 98.80% for people counting. Taking into account the positive results, the authors defend that the PWR system is applicable as it leverages existing commercial WiFi APs without requiring any modifications to the WiFi infrastructure or additional devices on the network. However, this advantage comes with added computational complexity, as the system relies completely on signal processing.

Wang *et al.* [22] approached the challenge of discontinuous communication in mobile devices, where smartphones suspend WiFi transmission to conserve battery, causing users to disappear temporarily from network scans. To maintain accuracy during these periods, the authors developed an event triggered framework that estimates occupancy based on entry and exit events. The framework also integrates a location filter using RSSI thresholds to discard devices located outside the target zone by applying a study of the mean values of the measured data in inside and outside positions which could be helpful for the current work, and a non-human filter to exclude stationary equipment, such as printers, via MAC address analysis. The authors also acknowledge limitations such as the reliance on RSSI thresholds restricts the system to zone level accuracy, the behavior of smartphones suspension of data transmission and the detection errors from user behavior, like occupants carrying multiple devices or separating from their smartphones.

In 2017, Ouf *et al.* [23] presented a case study to demonstrate the effectiveness of using WiFi to detect occupancy as opposed to the more common CO_2 sensors. To allow this comparison, the authors monitored WiFi connection counts and CO_2 at the same time, observing concentration levels in a university classroom over one week, using manual occupant counts as ground truth. The analysis revealed that WiFi counts served as a more suited predictor of occupancy, showing a statistical correlation of $r = 0.839$ when CO_2 levels show $r = 0.728$. Furthermore, the authors highlighted that while CO_2 sensors suffered from a detection lag of approximately 20 minutes and susceptibility to non-occupancy related fluctuations, WiFi data provided a more accurate, real time proxy for occupancy with the added benefit of utilizing existing infrastructure without additional cost.

In a Master's thesis, Verma [24] developed a framework for optimizing human resource allocation, directed for cleaning and maintenance, this was done by comparing three occupancy detection models: static university course schedules, thermal occupancy sensors, and WiFi location tracking. The study highlighted that while schedule based models are often inaccurate due to student absenteeism, thermal sensors provided the highest accuracy of approximately 98% by counting heat signatures at doorways. However, the author emphasized the negative considerations on the scalability of thermal sensors, due to a large installation cost. In contrast, the WiFi based model, utilizing the existing network achieved a comparable accuracy of roughly 90% with zero additional infrastructure costs. A supporting survey within the study validated the viability of this approach, revealing that 95.2% of campus users carried smartphones, with 90.5% actively logged into the university network, confirming that WiFi connection counts serve as a reliable, cost-effective proxy for real time occupancy.

In 2019, Nunna *et al.* [25] proposed a model for an occupancy monitoring system based on the RSSI detected by low-cost microcontrollers. To address the instability of wireless signals, the system employs a mean algorithm, which triggers an occupancy state only when the average RSSI over seven iterations dips below a calibrated threshold. Experimental trials demonstrated a high detection accuracy of 95%, approximately one error every 20 minutes, but only within a limited 3 meter line of sight range. The authors noted that beyond this 3 meter radius, the accuracy degrades due to multipath propagation. [1] This observation is relevant to the current thesis, as it highlights the physical limitations of RSSI based ranging in indoor environments.

3.2.2 Behavioral Monitoring

A case study performed by Christl [27] in 2024 analyzed the privacy implications of existing technologies for behavioral monitoring and profiling. The report examined solutions from major vendors such as Cisco, Juniper, Spacewell, and Locatee, identifying a trend where workplace infrastructure is used for tracking user behavior. Specifically in Section 3.2, the study focuses on Cisco, the manufacturer of the devices generating the logs utilized in this dissertation. The report shows a product called Cisco Spaces, a cloud platform that processes large amounts of data to profile user behavior. For user classification, the report notes that the system categorizes persons into groups like employee, student or guest based on the SSID category, this points that the system identifies the type of user based of the WiFi network the user connects to, allowing for distinct tracking of different user groups and their behaviors.

In 2016, Zhou *et al.* [28] proposed the EDUM (EDUcation Measurement) system to characterize educational behaviors using data collected from large-scale WiFi infrastructures, analyzing students punctuality based on longitudinal WiFi connection traces and to assess lecture attractiveness and student distraction. The system utilizes the mobile phone's interactive state, the screen on/off status, at a per minute basis. Deployed at Tsinghua University with approximately 700 student volunteers and 2,800 APs, the study provided results that revealed a negative correlation between high mobile phone usage during class and academic performance (GPA), and confirmed that students seated in the back rows exhibit significantly higher distraction levels than those in the front.

Later in 2025, Álvarez-Merino *et al.* [29] realized a study to investigate indoor location technologies for future integration in Smart Education (SE) environments, the authors reviewed several technologies, as mentioned earlier on this chapter, but for the sake of the thesis the focus will be toward WiFi. The authors identified WiFi as an accessible technology for SE, citing its availability in educational institutions and the connectivity provided by networks

[1]Phenomenon in wireless communication where a transmitted signal reaches the receiver antenna through multiple paths due to reflections and scattering from environmental objects, resulting in different signal copies with varying attenuation, phase shifts, and time delays as defined by Kaluuba *et al.* [26]

like eduroam, the one used on the present thesis. Regarding its capabilities, the study showed the potential of the IEEE 802.11mc standard, which allows for ranging, within approximately 1 meter, and preserves user privacy with calculations on the client device side. However, in their experimental PoC for automatic attendance, the WiFi approach achieved an accuracy of 93.77% using a regression model, it was outperformed by 5G (97.21%), leading the authors to conclude that while WiFi is a useful tool due to it's low cost and presence, a fusion of technologies provides the most robust solution for critical attendance monitoring.

3.2.3 Occupancy Monitoring

In the 2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS), Mohottige *et al.* [19] proposed a study to assess the feasibility of using WiFi AP infrastructure for room level occupancy monitoring across the University of New South Wales campus, a scenario similar to that of this thesis. The study had a duration of four weeks across rooms with varying numbers of APs, comparing WiFi data against beam counter sensors and ground truth enrollment numbers. To account for transient users, passing by, connecting to the APs, connections lasting less than five minutes were filtered out. The results indicated that the WiFi method achieved a stronger correlation with actual occupancy ($R = 0.85$) compared to the beam counters ($R = 0.68$).

Later in 2022, Mohottige *et al.* [30] published in the IEEE Sensors Journal a ML framework to infer classroom occupancy, sharing the same goal as the current work but employing standard statistical modeling. The authors utilized daily WiFi logs from the university's IT department, a scenario identical to this dissertation, comprising data from 70 APs including UU, MAC address, event timestamps, and AP names. Besides the similar data source, Mohottige *et al.* [30] identified critical limitations in using raw logs, such as the overlapping coverage of APs (where a student inside a room connects to a hallway AP), the presence of "bystanders", and the variability of multiple device connections per user. Ultimately, the framework achieved a symmetric Mean Absolute Percentage Error of 13.1%, a result comparable to dedicated beam counter sensors, this once more demonstrated that existing WiFi infrastructure can result in accurate occupancy estimation with no additional deployment costs, only computing costs.

In 2022, for a Master's thesis, Carrol-Woolery [31] proposed the "Building Floor Zone" technique to improve WiFi counting precision without requiring new hardware. The methodology involved mapping APs to specific zones based on room numbering and assigning hallway APs to the center of adjacent rooms, in similarity with the present thesis. Experimental analysis confirmed that this approach resulted in a higher statistical correlation with schedule based ground truth compared to standard floor level aggregation. An interesting mention is that the study found that counting all unique connections provided better accuracy than filtering for long duration sessions, challenging the assumption that transient users are merely noise.

Earlier in 2016, Anand *et al.* [32] proposed an attendance system that combines facial recognition for user authentication with WiFi network analysis to verify the student's location.

While the authors acknowledged methods like trilateration [2], due to signal interference adopted a fingerprinting approach using the ML algorithm. Regarding the results, the system demonstrated positive precision, achieving a positioning error between 1 and 2.5 meters. In practical testing, this methodology identified whether a student was inside the target classroom 94% of the time.

Khan *et al.* [33] presented a smartphone based attendance system utilizing WiFi signal strength for indoor localization. As noted in previous works, distinguishing user presence within confined boundaries, like a specific room, is challenging due to signal fluctuations. To address this, the authors employed a zone based fingerprinting approach, dividing the space into virtual grids and using machine learning to classify them. A valuable insight for this thesis is the authors explicit classification of signal quality, which categorizes RSSI values into ranges according to signal strength, something that could be necessary in the current work. The authors system achieved 100% accuracy in one test room and 92% in another. Furthermore, the study demonstrated significant robustness, the accuracy remained stable even when individual APs were removed or when different smartphone models were used.

3.2.4 Fingerprinting

In a study focusing on probabilistic localization, Le Dortz *et al.* [34] developed a WiFi fingerprinting system that utilizes probability distribution comparisons rather than simple RSSI averaging. Crucially for the context of this thesis, the methodology also used an offline phase to construct a radio map, a database where fingerprints were manually collected at several known rooms throughout the building. During the offline phase, the authors collected 100 RSSI samples at each reference point to estimate the signal strength probability distributions for every visible access point. This data collection process demonstrates the significant calibration effort, necessary in fingerprinting approaches. While their method achieved a median positioning error of 2.4 meters, the requirement to build and maintain such a detailed radio map highlights the scalability challenges that the passive, infrastructure based approach proposed in this current work seeks to eliminate.

Similarly, in 2020, Ninh *et al.* [35] proposed a random statistical method for indoor positioning that relies on the same two phase architecture common to fingerprinting solutions. The system explicitly distinguishes between an offline handling process, where a large number of WiFi signals are collected at specific reference points to create a database, and an online positioning process that uses the Mahalanobis distance to match live user data against this stored radio map.

3.3 Approaches to Privacy in Indoor Positioning Literature

The domain of privacy in data processing is made of several methodologies, ranging from approaches like k-anonymity to semantic definitions such as differential privacy. While these approaches are fundamental to the field, a comprehensive analysis of their details falls outside

[2]Trilateration is a geometric method of determining location by measuring distances from at least three known reference points using signal strength.

the scope of this dissertation. Instead, this section provides a high-level categorization of the privacy paradigms observed specifically within the context of indoor positioning literature. The objective is not to evaluate the cryptographic strength of these mechanisms, but to contextualize how current indoor positioning systems attempt to balance utility with user privacy, highlighting the limitations that motivate the adoption of a TEE approach.

The ubiquity of WiFi networks as a sensing infrastructure presents a fundamental problem, the same data that allows occupancy monitoring also introduces significant privacy risks for the individuals being tracked. As noted in the previous sections, transforming a standard wireless network into a soft sensor involves analyzing device footprints, specifically MAC addresses and signal characteristics, which can serve as strong identifiers of specific individuals.

Consequently, the academic literature demonstrates a variety of methodologies designed to balance the trade off between service utility and user privacy. These approaches range from data minimization strategies, that avoid collecting unique identifiers, to more complex anonymization schemes like hashing and k-anonymity. However, as the demand for room level precision increases, so does the inadequacy of traditional methods.

This section reviews these privacy preserving strategies, categorizing them into distinct paradigms found in the literature, omission of privacy mechanisms, where technical feasibility takes precedence, data aggregation and minimization, where sensitive data is aggregated at the source, anonymization and pseudonymization, where identifiers are masked, institutional compliance and ethical clearance, relying on governance frameworks and volunteer based or opt-in models.

3.3.1 Omission/Inexistence of Privacy Mechanisms

Wang *et al.* [22] focused on the technical application of detecting inactive smartphones to improve detection accuracy. While the system utilizes MAC address analysis to filter non-human devices, the study does not detail any mechanisms for anonymizing these addresses or protecting the identity of the smartphone owners.

Furthermore, Christl [27] provided an analysis of commercial indoor positioning systems, identifying a lack of privacy by design in several applications. The report shows that these platforms often prioritize profiling and categorizing users into groups like employees or students, over data protection, allowing for the tracking of individual movements without obfuscation or user optional mechanisms.

Mohottige *et al.* [19] conducted a validation of WiFi occupancy sensing by benchmarking it against hardware counters across a university campus. The study focused on the utility of the metadata, like user's MAC address which can endanger students privacy, since the authors did not detail privacy preserving architectures, it leads to the assumption of inexistence of privacy preserving mechanisms.

3.3.2 Data Aggregation and Minimization

Alishahi *et al.* [20] adopted a data minimization strategy by relying only on aggregated connection counts per AP. By converting raw network logs into numerical totals before analysis, the framework inherently discards individual identifiers, ensuring that no unique user data is retained.

Tang *et al.* [21] took the concept of data minimization to the physical layer by employing a device free approach known as Passive WiFi Radar. Instead of decoding data packets or logging MAC addresses, this method analyzes signal reflections caused by moving bodies. Consequently, the system avoids collecting any digital identifiers, making the data inherently anonymous and decoupling the occupancy detection from the users' personal devices.

Similarly, Ouf *et al.* [23] employed an aggregation strategy to validate WiFi sensing against environmental benchmarks. By utilizing the total number of connections as a bulk metric to correlate with CO_2 levels, the authors treated the crowd as a single entity. This approach avoids the privacy difficulties of tracking, as the system monitors the state of the room and not the behavior of the individuals within it.

Verma [24] demonstrated a resource optimization framework that relies on aggregated occupancy density. By only using bulk connection counts directly from the university's central IT department, rather than logging individual devices, the system calculates the "usage intensity" of classrooms.

3.3.3 Anonymization and Pseudonymization

Carrol-Woolery [31] implemented a form of pseudonymization to mitigate the risks of long-term profiling. In this study, the unique identifiers were masked using a hashing algorithm that was reset every 24 hours, something similar to what will be done during the development process of this dissertation.

3.3.4 Institutional Compliance and Ethical Clearance

Mohottige *et al.* [30] addressed the privacy implications of tracking students by operating under a governance framework, since the study obtained formal ethical clearance from the university.

3.3.5 Volunteer Based Model or Opt-in Model

Anand *et al.* [32] implemented a privacy model based on user interaction. In their system, attendance is not monitored passively, instead, students must actively engage with a smartphone application to capture a facial scan and submit their WiFi fingerprint, which falls in the category of an opt-in, ensuring that location data is only transmitted with the user's direct consent and knowledge.

Zhou *et al.* [28] employed a volunteer based model to justify the collection of highly granular behavioral data. In their "EDUM" system, the researchers recruited approximately 700 students who consented to having their WiFi traces and mobile application usage monitored.

Álvarez-Merino *et al.* [29] highlighted the privacy advantages of the IEEE 802.11mc standard, which shifts the location calculation to the client device. To demonstrate this architecture, the authors developed a mobile application, by requiring students to actively install and engage with this software to register their presence, the system uses an opt-in model.

Likewise, Khan *et al.* [33] developed a smartphone based attendance system that leverages RSSI zoning to verify student presence. Since the primary utility is the validation of attendance records using the students smartphone, the architecture operates on an opt-in model where students consent to the tracking of their devices.

3.4 Previous Works with SGX in Indoor Positioning

While the literature regarding the specific application of SGX to indoor positioning remains limited, a select number of studies have established a base foundation for this approach.

To address the privacy concerns associated with collecting user location data, Yan *et al.* [36] proposed a trusted computing framework based on SGX. The authors stated that traditional techniques like k-anonymity or encryption, may include computational overheads or require dependencies on a Trusted Third Party (TTP). In contrast, their work demonstrates that SGX allows a privacy scheme, by processing sensitive location data within a hardware protected enclave, the system ensures that raw identity information remains inaccessible to the operating system or the service provider, allowing secure analysis without exposing individual user data. This approach validates the architectural choice in this thesis to utilize SGX enclaves for processing sensitive WiFi logs, ensuring compliance with privacy standards while maintaining system performance.

Complementing the architectural advantages mentioned in the previous work, Kulkarni *et al.* [37] provided an evaluation of SGX for location based services, explicitly comparing it against traditional k-anonymity techniques. The study highlighted a trade off in privacy handling, that traditional obfuscation methods degrades service accuracy to achieve privacy. In contrast, their experiments demonstrated that an SGX based approach provides better result accuracy because the computation occurs on exact data within the secure enclave, protected from the host system. Furthermore, the authors quantified the performance cost of this security, finding that SGX introduces only a small penalty compared to bare metal implementation, due to the one time events such as the enclave creation, copying data, sealing and unsealing. This finding is important for the current thesis, as it validates that shifting the occupancy estimation logic into an SGX enclave is a viable strategy that secures student data without compromising the real time performance or the accuracy of the occupancy counts.

3.5 Data Processing in SGX

Processing large scale data within SGX introduces memory challenges, primarily due to the limited size of the EPC and the high cost of transitions between trusted and untrusted memory. Consequently, standard data processing frameworks cannot be directly ported to enclaves without significant performance overhead. To address these limitations, the state-of-the-art literature proposes various architectural strategies, ranging from data partitioning and streaming to hardware offloading and oblivious execution.

It is important to note that while some of the discussed frameworks utilize alternative trusted hardware, their architectural principles for mitigating memory constraints are applicable to SGX enclaves. The following subsections categorize these frameworks by their application domain, reviewing solutions for general-purpose processing, secure analytics, deep learning acceleration, and distributed federated learning.

3.5.1 General Purpose Data Processing

Schuster *et al.* [9] presented Verifiable Confidential Cloud Computing (VC3), a distributed MapReduce model [3] designed to address these limitations by strictly minimizing the TCB. Unlike systems that attempt to execute entire legacy applications or libraries from the OS inside an enclave, VC3 adopts a partitioned architecture, where the heavy lifting of job scheduling and data storage remains in the untrusted Hadoop (open source framework for distributed storage and processing of large datasets). Only the specific map and reduce functions, along with a minimal cryptographic shim, are loaded into the protected enclave. To cope with the memory constraints of the enclave, VC3 relies on a streaming data model. Encrypted data is ingested in input splits and processed within the enclaves heap, and the results are encrypted and then streamed out. To mitigate the performance overhead of SGX context switches, between `ECalls` and `OCalls`, VC3 utilizes a shared memory region for communication and implements aggressive batching. By processing key value pairs in batches, rather than individually, the system reduces the frequency of expensive enclave transitions. Furthermore, to avoid the memory footprint of standard libraries, VC3 uses a custom runtime library and heap allocator, ensuring that the limited secure memory is dedicated primarily to the user's data and logic.

In contrast to the streaming model employed by MapReduce frameworks, Priebe *et al.* [39] proposed EnclaveDB, a database engine designed to minimize the overhead of data processing by keeping all sensitive state resident within the enclave memory. Leveraging the Hekaton, a Structured Query Language (SQL) engine, EnclaveDB avoids the cost of disk I/O by ensuring that encryption and decryption occur only at the transaction boundaries rather than at the page level during execution. To reduce the TCB and optimize memory usage, the system adopts a pre-compiled execution model, complex query parsing and optimization are offloaded to a trusted client, and only the resulting native machine code is deployed to the

[3]Programming model for processing large data sets, users write map and reduce functions, and the execution of both functions is automatically parallelized and distributed, as defined by Dean and Ghemawat [38]

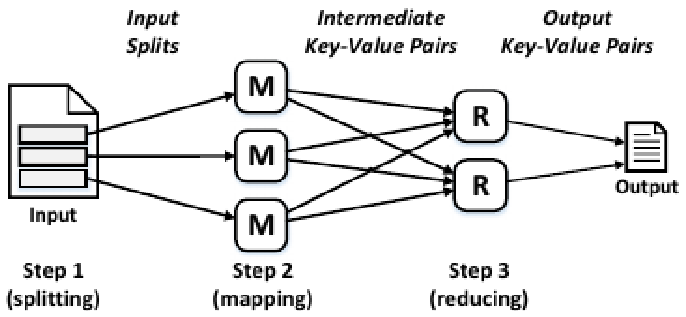

Figure 3.2: Steps of MapReduce with mappers (M) and reducers (R), image by Schuster *et al.* [9].

enclave. This architecture reduces the frequency of context switches, a known performance bottleneck in SGX, by executing entire transactions within the protected region without passing control back to the untrusted host until the transaction commits. Furthermore, to address the integrity of persistent data without the high memory cost of maintaining, a hash tree over the transaction log, EnclaveDB implements a lightweight protocol using hardware based counters and serialization points to guarantee the data updates and prevent rollback attacks.

3.5.2 Secure Analytics and Access Patterns

Advanced analytics and ML algorithms rely on matrix operations, however, executing these workloads within SGX introduces a vulnerability, standard algorithms show memory access patterns. These allow privileged adversaries to infer sensitive information from the execution trace, even when the memory contents are encrypted.

In 2017, Shaon *et al.* [40] introduced SGX-BigMatrix, a framework for performing secure matrix operations on encrypted data. Recognizing that large scale matrices often exceed the limited EPC capacity, the authors developed a BigMatrix abstraction that partitions data into fixed size, encrypted blocks. These blocks are dynamically loaded into the enclave for processing and evicted to untrusted memory via a software managed swapping mechanism, effectively bypassing the performance overhead of the operating system's paging. Crucially, to address the vulnerability of side-channel attacks, SGX-BigMatrix enforces data oblivious execution, by utilizing vectorized oblivious primitives and a specialized compiler to manage block transitions, the framework ensures that the sequence of memory accesses remains independent of the sensitive data, preventing adversaries from inferring information through memory access traces.

In the domain of privacy-preserving ML, Ohrimenko *et al.* [41] addressed the challenge of side-channel leakages in SGX-based data processing. The authors stated that although SGX allows for memory confidentiality, standard ML algorithms show data-dependent memory access patterns that can be observed by a privileged attacker to infer sensitive information, to mitigate this, they proposed data-oblivious for several ML algorithms. Instead of standard conditional branches, which leak information through the instruction trace, the authors implementation utilizes oblivious primitives based on conditional move instructions and vectorized memory operations. This ensures that the execution trace depends solely on public parameters, such as the dataset size, rather than the private input data. Their evaluation demonstrated that this hardware-assisted approach is significantly more efficient than purely cryptographic alternatives, such as homomorphic encryption, enabling the training of models on realistic datasets with manageable overhead.

Complementing the discussion on performance trade-offs by Ohrimenko *et al.* [41], Chandra *et al.* [42] argued that data-oblivious techniques, while adding security guarantees, often lead to performance overheads for analytics. To address this issue, the authors proposed a randomization-based defense strategy, to do this, instead of eliminating data-dependent access patterns, which requires expensive actions, their approach adds noise into the memory access traces. Specifically, the noise is introduced with the generation of dummy data instances that mimic the valid dataset, these are created by selecting random feature values uniformly distributed within the ranges from the original data, ensuring they possess no distinguishing characteristics. These dummy records are then shuffled with the real user data using a sorting network based on random keys generated via SGX's hardware random number generator function. By masking the true data access sequence with random memory operations, this method prevents side-channel attackers from inferring sensitive information with high confidence. This introduces a strategic trade-off between privacy and efficiency, with the strict determinism of data-obliviousness in favor of a probabilistic protection model, the system achieves significantly higher computational throughput, making it more suitable for large-scale data analytics tasks.

3.5.3 Deep Learning and Hardware Acceleration

The deployment of Deep Neural Networks (DNN)s in trusted environments faces the difficulty of the conflict between the computational demands of deep learning and the hardware limitations of standard enclaves. SGX is CPU-bound and lacks direct access to trusted accelerators, such as Graphics Processing Unit (GPU)s, creating a bottleneck for matrix operations. Furthermore, the memory footprint of modern models and intermediate activations exceeds the trusted capacity of the EPC.

Addressing the hardware limitations of executing neural networks within CPU-bound enclaves, in 2019 Tramèr and Boneh [43] proposed Slalom, a framework for verifiable and private execution. Recognizing that the lack of trusted GPU support in SGX creates a significant bottleneck for heavy matrix operations, the authors introduced a hybrid execution model. In this approach, the enclave delegates work to a remote or local GPU to use its processing power, but this delegation does not align with the confidential necessities, so to ensure security, it uses a verification mechanism based on Freivalds' algorithm, that allows the TEE to check the integrity of the GPU's output without re-executing the full computation. For privacy-preserving inference, the system utilizes input blinding techniques, ensuring that the data processed by the untrusted GPU remains opaque, this decouples the security logic from the bulk computation, demonstrating a viable path for high-performance deep learning in confidential computing.

Addressing data processing under severe memory constraints, Mo *et al.* [44] proposed DarkneTZ, a framework designed to execute DNN within the limited trusted memory of edge devices. Although designed for ARM TrustZone on edge devices rather than Intel SGX, their architectural approach is relevant to solving EPC limitations. Instead of processing the entire computation inside the TEE, DarkneTZ employs a model partitioning strategy. The system splits the data flow, the initial, expensive feature extraction layers are processed in the untrusted environment, while the intermediate activations are securely transferred to the TEE for the final classification layers. Within the enclave, the processing logic includes a data sanitization step, by normalizing the output confidence scores, before returning them to the untrusted OS.

To address the bottleneck of processing DNN training workloads within trusted hardware, Asvadishirehjini *et al.* [45] proposed GOAT, a system that restructures the data processing pipeline through secure outsourcing. Instead of processing the entire dataset within the enclave, GOAT delegates the matrix operations of forward and backward propagation to an untrusted GPU. To maintain data integrity during this external processing, the system employs an asynchronous probabilistic verification strategy, the TEE acts as an auditor, selecting data batches to re-process internally. By comparing these trusted computations against the GPU's outputs in parallel with the main training loop, the system statistically guarantees the correctness of the outsourced processing without stalling the data flow, bridging the gap between the integrity of TEEs and the throughput of hardware accelerators.

Expanding the scope of secure data processing beyond fixed CPU instruction sets, Oh *et al.* [46] introduced MeetGo, a TEE architecture designed for Field-Programmable Gate Arrays (FPGA)s. In this paradigm, data processing is not defined by software instructions but by reconfigurable hardware circuits. MeetGo utilizes Dynamic Partial Reconfiguration (DPR) to instantiate isolated hardware modules, like the enclave, on demand within the FPGA fabric. The processing pipeline is controlled by a hardware-based security manager, which performs line-rate decryption and encryption of data streams entering and leaving the processing modules. This architecture allows for high-throughput, custom hardware acceleration of sensitive workloads while enforcing memory isolation through specialized hardware monitors, effectively preventing the user-defined logic from accessing unauthorized memory regions during execution, but this comes with the additional cost of the FPGA hardware.

While frameworks like DarkneTZ address memory limitations by offloading layers to untrusted hardware, Truong *et al.* [47] proposed optimization techniques designed to keep the inference execution within the enclave. Identifying that the standard execution flow of convolutional layers triggers page thrashing, the authors introduced y-plane partitioning. This strategy divides the input and output tensors into horizontal bands that fit within the enclave's secure memory, ensuring that the working set remains resident in the secure cache.

3.5.4 Distributed and Federated Learning

In the context of federated learning, where a central server aggregates model updates from numerous remote clients, memory constraints become even more noticeable due to the high dimensionality of modern models.

Xu *et al.* [48] proposed SGX-FL to address the performance bottleneck of aggregating gradient vectors within SGX by modifying the data processing logic to exploit sparsity. Instead of processing full dense updates, which often trigger severe page thrashing in the EPC, the authors implemented a Top-k gradient selection algorithm directly within the enclave. This technique filters incoming data, aggregating only the most significant gradients while discarding negligible values, thereby drastically reducing the memory working set. Furthermore, the system optimizes the remaining computation through block-wise data loading and double buffering, ensuring that the processing of gradient blocks aligns with the CPU cache size to minimize memory access latency during the secure aggregation phase.

Scaling data processing beyond the limits of a single trusted enclave, Quoc *et al.* [49] introduced SecureTF, a mechanism for distributed machine learning execution. This mechanism partitions the data processing workflow into a parameter servers architecture, where the computational load is distributed across multiple enclaves, workers compute gradients on private data, while parameter servers aggregate these updates to refine the global model. To handle the intensive memory demands of tensor operations that typically exceed the trusted memory of the EPC, the system uses a shielded execution runtime, in this case SCONE [4] , to manage secure paging and encrypted I/O transparently during execution. This allows for the stateful processing of large-scale datasets and complex computation graphs across a cluster of TEEs, ensuring that data remains encrypted not only at rest but also throughout the entire distributed calculation pipeline.

In the area of federated learning, secure aggregation typically relies on cryptographic operations such as encryption, which introduces computational overhead. Mo *et al.* [51] proposed PPFL, a framework that shifts the processing from cryptographic computation to hardware-based isolation. By utilizing the TEE as a trusted aggregator, the system performs model updates in plaintext within the secure memory, avoiding the performance penalties of operations on ciphertext. To optimize the data flow against network latency and client variability, PPFL implements a greedy grouping strategy within the aggregation logic, this mechanism batches and processes incoming gradients from clients without waiting for stragglers, streamlining the training pipeline and reducing the total convergence time compared to cryptographic approaches.

Mondal *et al.* [52] proposed FLATEE, a framework that leverages TEEs to replace expensive cryptographic protocols in federated learning. By performing model aggregation within enclaves at the server side and incorporating several algorithms for data aggregation and resilience, the system ensures privacy and resilience against poisoning attacks while reducing training and communication latency compared to traditional crypto-based approaches.

Addressing the vulnerability of gradient linkability, Zhang *et al.* [53] introduced a shuffling mechanism within the TEE processing pipeline by the name of ShuffleFL. Recognizing that the sequence of data arrival and memory access patterns can reveal the source of a model, even when the payload is encrypted, the proposed system includes a randomization layer before aggregation. Incoming gradients are buffered and subjected to a random permutation inside the enclave, decoupling the gradient values from their identifiers, ensuring unlinkability, preventing the server from correlating specific model updates with their originating clients based on communication metadata or execution timing.

Kato *et al.* [54] identified that data-dependent optimizations introduce side-channel vulnerabilities. Processing sparse gradients reveals memory access patterns, specifically, the indices of non-zero updates, which can allow an adversary to infer feature distributions. To address this, the authors proposed Olive, a framework that reconciles efficiency with security through oblivious sparse aggregation, so instead of directly accessing sparse indices, the system

[4]A secure container mechanism for Docker that uses the SGX trusted execution support of Intel CPUs to protect container processes from outside attacks, by Arnautov *et al.* [50].

employs hardware oblivious primitives to align and aggregate gradients. By ensuring that the memory access trace remains independent of the actual values and indices of the gradient updates, it closes the leakage channel associated with data processing, with the additional cost of reintroducing some computational overhead compared to simple sparse aggregation.

Brossard *et al.* [55] introduced Veracruz, a framework designed to allow secure delegated computations. The central feature of this architecture is the utilization of a WebAssembly environment within the enclave, which allows the execution of arbitrary programs defined in various languages. This design establishes a dual layer, combining the hardware-enforced protection of the SGX enclave with the software-based sandboxing of the Wasm runtime. Functionally, the system acts as a trusted component on the untrusted host, enabling multiple mutually independent parties to provision both input data and computational modules via remotely attested channels. By relying on the runtime binary format, the framework abstracts the hardware complexity and ensures that the data processing logic strictly confined within the secure boundary.

3.6 Summary

The review of the literature reveals a gap in the intersection of indoor positioning and privacy-preserving technologies. The dependency on logs introduces challenges that require the usage of filters, as identified in the reviewed literature, occupancy is affected by stationary devices, transient users or bystanders who are passing by the access point's range, and discontinuous communications from mobile devices entering power-saving modes.

Although literature validates WiFi connection logs as an effective and scalable proxy for occupancy counting, reviewed implementations face a dilemma, overlook user privacy to prioritize utility or rely on opt-in models that introduce scalability and accuracy issues.

TEE, such as Intel SGX, appears as a possible solution to this limitation, allowing for the processing of sensitive data without exposing it to the infrastructure. However, SGX in large-scale data processing introduces challenges, due to the limited EPC size and the overhead of context switching between trusted and untrusted environments.

While frameworks like VC3 and EnclaveDB address large-scale data processing, and recent systems such as Slalom, DarkneTZ, or PPFL focus on heavy ML workloads and federated aggregation, they introduce complexity and hardware dependencies. These approaches are not suited for the objectives of this dissertation, which requires a lightweight monitoring system capable of handling continuous streams of WiFi logs with minimal overhead.

Consequently, this thesis proposes an architecture that uses the security mechanisms of SGX but implements a custom memory management strategy.

References

[1] M. Sabt, M. Achemlal, and A. Bouabdallah, «Trusted execution environment: What it is, and what it is not», vol. 1, pp. 57–64, 2015. DOI: 10.1109/Trustcom.2015.357.

[2] J. González, «Operating system support for run-time security with a trusted execution environment», Mar. 2015. DOI: 10.13140/RG.2.1.4827.8161.

[3] S. Pinto and N. Santos, «Demystifying arm trustzone: A comprehensive survey», *ACM Comput. Surv.*, vol. 51, no. 6, Jan. 2019, ISSN: 0360-0300. DOI: 10.1145/3291047. [Online]. Available: https://doi.org/10.1145/3291047.

[4] M. Misono, D. Stavrakakis, N. Santos, and P. Bhatotia, «Confidential vms explained: An empirical analysis of amd sev-snp and intel tdx», *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 3, Dec. 2024. DOI: 10.1145/3700418. [Online]. Available: https://doi.org/10.1145/3700418.

[5] V. Costan and S. Devadas, «Intel sgx explained»,

[6] T. Silva and A. Zúquete, «Ambientes de execução segura», Univerity of Aveiro, Tech. Rep., 2025.

[7] P.-C. Cheng, W. Ozga, E. Valdez, *et al.*, «Intel tdx demystified: A top-down approach», *ACM Comput. Surv.*, vol. 56, no. 9, Apr. 2024, ISSN: 0360-0300. DOI: 10.1145/3652597. [Online]. Available: https://doi.org/10.1145/3652597.

[8] C. T. Li, J. C. Cheng, and K. Chen, «Top 10 technologies for indoor positioning on construction sites», *Automation in Construction*, vol. 118, p. 103 309, 2020, ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2020.103309. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580519306016.

[9] F. Schuster, M. Costa, C. Fournet, *et al.*, «Vc3: Trustworthy data analytics in the cloud using sgx», pp. 38–54, 2015. DOI: 10.1109/SP.2015.10.

[10] P. de Keyser, «8 - metadata formats and indexing», in *Indexing*, ser. Chandos Information Professional Series, P. de Keyser, Ed., Chandos Publishing, 2012, pp. 143–166, ISBN: 978-1-84334-292-2. DOI: https://doi.org/10.1016/B978-1-84334-292-2.50008-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781843342922500088.

[11] P. Quinn, «The difficulty of defining sensitive data – the concept of sensitive data in the eu data protection framework», *SSRN Electronic Journal*, Jan. 2020. DOI: 10.2139/ssrn.3713134.

[12] Global eduroam Governance Committee, «eduroam Compliance Statement», GÉANT, Compliance Statement, version v2-FINAL, n.d. [Online]. Available: https://eduroam.org/.

[13] A. Paju, M. O. Javed, J. Nurmi, J. Savimäki, B. McGillion, and B. B. Brumley, «Sok: A systematic review of tee usage for developing trusted applications», in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, ser. ARES '23, Benevento, Italy: Association for Computing Machinery, 2023, ISBN: 9798400707728. DOI: 10.1145/3600160.3600169. [Online]. Available: https://doi.org/10.1145/3600160.3600169.

[14] D. Kaplan, «Hardware vm isolation in the cloud», *Commun. ACM*, vol. 67, no. 1, pp. 54–59, Dec. 2023, ISSN: 0001-0782. DOI: 10.1145/3624576. [Online]. Available: https://doi.org/10.1145/3624576.

[15] Q. Do, B. Martini, and K.-K. R. Choo, «The role of the adversary model in applied security research», *Computers and Security*, vol. 81, pp. 156–181, 2019, ISSN: 0167-4048. DOI: https://doi.org/10.1016/

j.cose.2018.12.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818306369.

[16] H. Zou, H. Jiang, J. Yang, L. Xie, and C. Spanos, «Non-intrusive occupancy sensing in commercial buildings», *Energy and Buildings*, vol. 154, pp. 633–643, 2017, ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2017.08.045. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778816311987.

[17] W. Wang, J. Wang, J. Chen, G. Huang, and X. Guo, «Multi-zone outdoor air coordination through wi-fi probe-based occupancy sensing», *Energy and Buildings*, vol. 159, pp. 495–507, 2018, ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2017.11.041. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778817321217.

[18] I. Sobron, J. Del Ser, I. Eizmendi, and M. Vélez, «Device-free people counting in iot environments: New insights, results, and open challenges», *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4396–4408, 2018. DOI: 10.1109/JIOT.2018.2806990.

[19] I. P. Mohottige, T. Sutjarittham, N. Raju, H. H. Gharakheili, and V. Sivaraman, «Role of campus wifi infrastructure for occupancy monitoring in a large university», pp. 1–5, 2018. DOI: 10.1109/ICIAFS.2018.8913341.

[20] N. Alishahi, M. Nik-Bakht, and M. M. Ouf, «A framework to identify key occupancy indicators for optimizing building operation using wifi connection count data», *Building and Environment*, vol. 200, p. 107 936, 2021, ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2021.107936. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132321003401.

[21] C. Tang, W. Li, S. Vishwakarma, K. Chetty, S. Julier, and K. Woodbridge, «Occupancy detection and people counting using wifi passive radar», pp. 1–6, 2020. DOI: 10.1109/RadarConf2043947.2020.9266493.

[22] J. Wang, N. C. F. Tse, and J. Y. C. Chan, «Wi-fi based occupancy detection in a complex indoor space under discontinuous wireless communication: A robust filtering based on event-triggered updating», *Building and Environment*, vol. 151, pp. 228–239, 2019, ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2019.01.043. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132319300794.

[23] J. M. Ouf, M. H. Issa, A. Azzouz, and A.-M. Sadick, «Effectiveness of using wifi technologies to detect and predict building occupancy», *Research Gate*, vol. 2, 2017. DOI: https://doi.org/10.1051/sbuild/2017005.

[24] P. Verma, «Classroom occupancy-based human resource optimization using sensor- and wifi-based location tracking», 2017. DOI: 10.7939/R3HT2GQ36. [Online]. Available: https://doi.org/10.7939/R3HT2GQ36.

[25] A. Nunna, A. Varma, R. Kumar N., S. Tarun, and M. Sangeetha, «Classroom automation using rssi», pp. 1–6, 2019. DOI: 10.1109/ICESIP46348.2019.8938387.

[26] L. Kaluuba, G. Taban-Wani, and D. Waigumbulizi, «The fading channel problem and its impact on wireless communication systems in uganda», J. Mwakali and G. Taban-Wani, Eds., pp. 621–634, 2006. DOI: https://doi.org/10.1016/B978-008045312-5/50068-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080453125500686.

[27] W. Christl, «Tracking indoor location, movement and desk occupancy in the workplace», *Research Gate*, 2019. DOI: https://doi.org/10.1109/ISECon.2019.8882085.

[28] M. Zhou, M. Ma, Y. Zhang, K. SuiA, D. Pei, and T. Moscibroda, «Edum: Classroom education measurements via large-scale wifi networks», UbiComp '16, pp. 316–327, 2016. DOI: 10.1145/2971648.2971657. [Online]. Available: https://doi.org/10.1145/2971648.2971657.

[29] C. S. Álvarez-Merino, E. J. Khatib, A. T. Muñoz, and R. B. Moreno, «Integrating indoor localization technologies for enhanced smart education: Challenges, innovations, and applications», *IEEE Access*, vol. 13, pp. 105 317–105 333, 2025. DOI: 10.1109/ACCESS.2025.3578718.

[30] I. P. Mohottige, H. H. Gharakheili, T. Moors, and V. Sivaraman, «Modeling classroom occupancy using data of wifi infrastructure in a university campus», *IEEE Sensors Journal*, vol. 22, no. 10, pp. 9981–9996, 2022. DOI: `10.1109/JSEN.2022.3165138`.

[31] L. Carrol-Woolery, «Localization and counting of indoor populations on a university campus using wi-fi connection logs and floor plans», 2022. [Online]. Available: `http://hdl.handle.net/10012/19281`.

[32] S. Anand, K. Bijlani, S. Suresh, and P. Praphul, «Attendance monitoring in classroom using smartphone and wi-fi fingerprinting», *2016 IEEE Eighth International Conference on Technology for Education*, pp. 62–67, 2016. DOI: `10.1109/T4E.2016.021`.

[33] S. M. Khan, M. T. Maliha, M. S. Haque, and A. Rahman, «Wifi received signal strength (rss) based automated attendance system for educational institutions», pp. 172–180, 2025. DOI: `10.1145/3704522.3704523`. [Online]. Available: `https://doi.org/10.1145/3704522.3704523`.

[34] N. Le Dortz, F. Gain, and P. Zetterberg, «Wifi fingerprint indoor positioning system using probability distribution comparison», in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 2301–2304. DOI: `10.1109/ICASSP.2012.6288374`.

[35] D. B. Ninh, J. He, V. T. Trung, and D. P. Huy, «An effective random statistical method for indoor positioning system using wifi fingerprinting», *Future Generation Computer Systems*, vol. 109, pp. 238–248, 2020, ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2020.03.043`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0167739X19324835`.

[36] Z. Yan, X. Qian, S. Liu, and R. Deng, «Privacy protection in 5g positioning and location-based services based on sgx», *ACM Trans. Sen. Netw.*, vol. 18, no. 3, Aug. 2022, ISSN: 1550-4859. DOI: `10.1145/3512892`. [Online]. Available: `https://doi.org/10.1145/3512892`.

[37] V. Kulkarni, B. Chapuis, and B. Garbinato, «Privacy-preserving location-based services by using intel sgx», *Proceedings of the First International Workshop on Human-centered Sensing, Networking, and Systems (HumanSys'17)*, no. CONFERENCE, 6 p. CHAPUIS, Bertil est chercheur à la HES-SO, HEIG-VD, depuis 2021. DOI: `https://doi.org/10.1145/3144730.3144739`. [Online]. Available: `http://arodes.hes-so.ch/record/15319`.

[38] J. Dean and S. Ghemawat, «Mapreduce: Simplified data processing on large clusters», *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008, ISSN: 0001-0782. DOI: `10.1145/1327452.1327492`. [Online]. Available: `https://doi.org/10.1145/1327452.1327492`.

[39] C. Priebe, K. Vaswani, and M. Costa, «Enclavedb: A secure database using sgx», pp. 264–278, 2018. DOI: `10.1109/SP.2018.00025`.

[40] F. Shaon, M. Kantarcioglu, Z. Lin, and L. Khan, «Sgx-bigmatrix: A practical encrypted data analytic framework with trusted processors», CCS '17, pp. 1211–1228, 2017. DOI: `10.1145/3133956.3134095`. [Online]. Available: `https://doi.org/10.1145/3133956.3134095`.

[41] O. Ohrimenko, F. Schuster, C. Fournet, *et al.*, «Oblivious multi-party machine learning on trusted processors», in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16, Austin, TX, USA: USENIX Association, 2016, pp. 619–636, ISBN: 9781931971324.

[42] S. Chandra, V. Karande, Z. Lin, L. Khan, M. Kantarcioglu, and B. Thuraisingham, «Securing data analytics on sgx with randomization», in *Computer Security – ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds., Cham: Springer International Publishing, 2017, pp. 352–369, ISBN: 978-3-319-66402-6.

[43] F. Tramèr and D. Boneh, *Slalom: Fast, verifiable and private execution of neural networks in trusted hardware*, 2019. arXiv: `1806.03287 [stat.ML]`. [Online]. Available: `https://arxiv.org/abs/1806.03287`.

[44] F. Mo, A. S. Shamsabadi, K. Katevas, *et al.*, «Darknetz: Towards model privacy at the edge using trusted execution environments», in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '20, Toronto, Ontario, Canada: Association for Computing Machinery, 2020, pp. 161–174, ISBN: 9781450379540. DOI: `10.1145/3386901.3388946`. [Online]. Available: `https://doi.org/10.1145/3386901.3388946`.

[45] A. Asvadishirehjini, M. Kantarcioglu, and B. Malin, *Goat: Gpu outsourcing of deep learning training with asynchronous probabilistic integrity verification inside trusted execution environment*, 2020. arXiv: `2010.08855 [cs.CR]`. [Online]. Available: `https://arxiv.org/abs/2010.08855`.

[46] H. Oh, K. Nam, S. Jeon, Y. Cho, and Y. Paek, «Meetgo: A trusted execution environment for remote applications on fpga», *IEEE Access*, vol. 9, pp. 51 313–51 324, 2021. DOI: `10.1109/ACCESS.2021.3069223`.

[47] J.-B. Truong, W. Gallagher, T. Guo, and R. J. Walls, «Memory-efficient deep learning inference in trusted execution environments», in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 161–167. DOI: `10.1109/IC2E52221.2021.00031`.

[48] T. Xu, K. Zhu, A. Andrzejak, and L. Zhang, «Distributed learning in trusted execution environment: A case study of federated learning in sgx», in *2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*, 2021, pp. 450–454. DOI: `10.1109/IC-NIDC54101.2021.9660433`.

[49] D. L. Quoc, F. Gregor, S. Arnautov, R. Kunkel, P. Bhatotia, and C. Fetzer, «Securetf: A secure tensorflow framework», in *Proceedings of the 21st International Middleware Conference*, ser. Middleware '20, Delft, Netherlands: Association for Computing Machinery, 2020, pp. 44–59, ISBN: 9781450381536. DOI: `10.1145/3423211.3425687`. [Online]. Available: `https://doi.org/10.1145/3423211.3425687`.

[50] S. Arnautov, B. Trach, F. Gregor, *et al.*, «SCONE: Secure linux containers with intel SGX», in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Savannah, GA: USENIX Association, Nov. 2016, pp. 689–703, ISBN: 978-1-931971-33-1. [Online]. Available: `https://www.usenix.org/conference/osdi16/technical-sessions/presentation/arnautov`.

[51] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, «Ppfl: Privacy-preserving federated learning with trusted execution environments», in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '21, Virtual Event, Wisconsin: Association for Computing Machinery, 2021, pp. 94–108, ISBN: 9781450384438. DOI: `10.1145/3458864.3466628`. [Online]. Available: `https://doi.org/10.1145/3458864.3466628`.

[52] A. Mondal, Y. More, R. H. Rooparaghunath, and D. Gupta, «Flatee: Federated learning across trusted execution environments», *CoRR*, vol. abs/2111.06867, 2021. arXiv: `2111.06867`. [Online]. Available: `https://arxiv.org/abs/2111.06867`.

[53] Y. Zhang, Z. Wang, J. Cao, R. Hou, and D. Meng, «Shufflefl: Gradient-preserving federated learning using trusted execution environment», in *Proceedings of the 18th ACM International Conference on Computing Frontiers*, ser. CF '21, Virtual Event, Italy: Association for Computing Machinery, 2021, pp. 161–168, ISBN: 9781450384049. DOI: `10.1145/3457388.3458665`. [Online]. Available: `https://doi.org/10.1145/3457388.3458665`.

[54] F. Kato, Y. Cao, and M. Yoshikawa, «Olive: Oblivious and differentially private federated learning on trusted execution environment», Feb. 2022. DOI: `10.48550/arXiv.2202.07165`.

[55] M. Brossard, G. Bryant, B. E. Gaabouri, *et al.*, «Private delegated computations using strong isolation», *IEEE Transactions on Emerging Topics in Computing*, vol. 12, no. 1, pp. 386–398, 2024. DOI: `10.1109/TETC.2023.3281738`.

APPENDIX A

Additional content