Università
Ca'Foscari
Venezia

Master's Degree
in Computer Science
Software Dependability and Cyber Security

Final Thesis

An Area of Trust - Trusted Execution Environment

Supervisor:
Prof. Focardi Riccardo

Regonda Brahmashwini
887689

2021/2022

An Area of Trust – Trusted Execution Environment

Brahmashwini Regonda

Abstract

Nowadays, aiming to design complex and safer systems has become a necessity. For this reason, TEEs (Trusted Execution Environments) has been introduced. TEEs have become increasingly popular in present-day CPUs as they allow a way to execute hardware-supported security services. TEEs are in fact used to improve data security. However, due to security vulnerabilities, they have become a target of various attacks.

Intel SGX is the most implemented and tested hardware-based encryption TEE. This study provides a general overview of the features, benefits, limitations, and vulnerabilities of TEEs based on Intel SGX hardware and other architectures, analyzing some attack mitigations. Some generic Intel SGX server-side features and applications have been described such as data sealing, attestation, data privacy, data protection, and encryption.

The results of this study show that using a few SGX techniques can stop some kinds of attacks, but side-channel attacks cannot be completely prevented.

Keywords: Intel SGX, Attacks, Isolation, Security

Acknowledgment

I would like to express special thanks to my supervisor Professor Focardi Riccardo, you have been a great mentor for me.

Secondly, I wanted to take this opportunity to thank my parents my husband "Regonda Srihari, Regonda Laxmi, and my husband Thodupunoori Vinay Kumar "for giving me all the support mentally and guiding me in making my masters a successful experience.

I am happy that some of my friends helped me in achieving my goal and would like to thank all of them for their contributions.

Finally Thank God for the opportunity and the love he showed towards me for a good future ahead.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

TEE	Trusted Execution Environments
SGX	Software Guard Extension
EDL	Enclave Definition Language
ECALLs	Enclave Calls
OCALLs	Outside Calls
EPC	Enclave Page Cache
DRAM	Dynamic Random Access Memory
PRM	Processor Reserved Memory
EPCM	Enclave Page Cache Map
AMD	Advanced Micro Devices
ARM	Advanced RISC Machine
TPM	Trusted Platform Module

TCB	Trusted Computing Base
BIOS	Basic Input/Output System
ECDH	Elliptic Curve Diffie Hellman
AES	Advanced Encryption Standard
ISV	Independent Software Vendor
AES-NI	Advanced Encryption Standard New Instructions
SEV	Secure Encrypted Virtualization
SME	Secure Memory Encryption
MITM	Man in The Middle
LLC	Last Level Cache
PTE	Page Table Entry
IBM	International Business Machines Corporation
IAM	Identity and Access Management
SQL	Structured Query Language

IoT	Internet of Things
ME	Management Engine
SCAs	Side Channel Attacks
GB	Giga Byte
DoS	Denial of Service
OS	Operating System
CPU	Central Processing Unit
ASLR	Address Space Layout Randomization
PDP	Provable Data Possession
TPA	Third Party Auditor
SSL	Secure Sockets Layer
TPM	Trusted Platform Module
DRM	Digital Rights Management

SECS	SGX Enclave Control Structure
PII	Personally Identifiable Information
SSL	Secure Sockets Layer

Chapter 1

1. Introduction

User necessity for security is increasing as social and economic development has created new demands and encouraged the development of new expertise in cyberspace. Communications between modern systems are becoming more complex as technology is developing like parallel and multi-processes environments which have distinct security and trust levels running in parallel for both software and hardware sources. Security-sensitive applications run on an untrusted computing base including OS, firmware, and client applications. Although conventional protection methods, like complete homomorphic encryption [36] and security perimeter techniques cannot fully help the rising security needs of present-day systems [37]. Robustifying the security of the modern complex system is a crucial thing nowadays.

Trusted computing was described to assist systems to get secure computation, data protection, and privacy [40]. This is an interesting strive to improve the security of modern devices by lowering the credence of the hardware and software TCB [38]. This performs a safe computation on a system that is maintained by a third party. The TPM was an option to gain remote trusted computing in starting days. This separates the hardware part that allows services like sealing data, remote and local attestation, and managing cryptographic keys. Although it is not dynamic and does not support memory protection. For mitigating this drawback, a good approach was proposed that allows code to run in an isolated environment to provide confidentiality along with integrity. This is called Trusted Execution Environment (TEE).

Up to now many producers have manufactured their TEEs like Intel ME [41], AMD platform Secure Processor, ARM TrustZone [10, 25, 40], Intel SGX, and AMD Memory Encryption [4, 42]. In all these products Intel SGX is one of the characteristic and uncommon TEE. This was implemented to improve the protection of Intel CPU products like servers, and desktops.

SGX has sealing data, memory isolation, and local & remote attestation these 3 are important security features. Data sealing secures enclave data during execution. Memory isolation prevents enclave memory from modifying and accessing in an unauthorized way during runtime. An attestation is a mechanism where cryptographic keys are utilized to publish its operations for the configured software of the enclave and then verified locally or remotely. In addition to these three features, SGX is utilized effectively to seal the data and operations like encryption and signing within an isolated environment so that malicious entities should not get a permit to crucial information directly.

Currently, Intel SGX enclaves provide a secure environment to protect sensitive programs on CPUs in the presence of a malicious entity. SGX is mostly used to improve the security of systems [2, 3, 12, 14, 25, 26, 28, 29, 43, 44, 45, 46, 47, 48, 49,50,52,63,65]. Also, no far direct attacks on SGX have been announced or recorded. But there are a few serious security flaws and threats that can be resolved and are yet to be discovered. Although the number of works tells that SGX is weak towards many types of SCAs [13, 17, 34, 51, 52, 53, 54, 55, 58, 60, 71, 73, 74, 75] and speculative attacks [56, 57].

1.1 Outline

The thesis is structured as follows:

Chapter 1 describes the problem statement of the thesis, related work, literature review, main contributions, background, and motivation of my thesis.

Chapter 2 describes the secure enclave, enclaves debug mode, memory organization of the enclave, enclave ECALLs & OCALLS, and hardware technologies like ARM Trust zone, Intel SGX, AMD SEV, AMD SEV vs Intel SGX is described. And two use cases of TEEs

Chapter 3 contains a detailed description of Intel SGX technology. How to enable Intel SGX, and features of Intel SGX like data sealing, local and remote attestation. And an analysis of Intel SGX.

Chapter 4 first describes the benefits and limitations of TEE. And then details the vulnerabilities such as side channels, software vulnerabilities, speculative execution of SGX, and protection methods from the SCAs. Also tells about Randomization and Obfuscation techniques which help the hardening of SGX and prevention from a few attacks.

Chapter 5 discusses client-side and server-side TEE. Firstly, on the client side, it reports about the attacks on a password manager and protection for the client side using Intel SGX and some protection without Intel SGX. Secondly Intel SGX for server-side and applications of Intel SGX for server-side like data-in-use protection, data privacy, and integrity of cloud data.

Chapter 6

6.1 Conclusions summarize the thesis work

6.2 Future work describes four research problems and challenges which might help in implementing SGX more securely in the future.

1.2 Problem Statement

The development process and conceptual background of the Intel SGX technology are very difficult and complex to get started. So, if developers lack the knowledge about the weakness of SGX then it may lead to breaches. And so, to develop an application securely using Intel SGX the person who develops should have a solid understanding of the technology and what makes security guarantees that SGX can provide to their applications. This thesis guides the security features, applications, and attacks of Intel SGX so that it can be used by developers to develop and program applications more robustly.

A) What are the security features of Intel SGX that helps to protect data?

B) What are some applications of Intel SGX and TEEs?

C) Why SGX is vulnerable?

D) What are the mitigations of side-channel attacks?

SGX is still a new technology and not many applications make use of this. This helps developers to explore the way SGX can be utilized to create a safe application and protection on the client-side and server-side from cyber-attacks. This would also provide a reader with the applications of SGX.

1.3 Background and Motivation

In today’s real world the need for security of the user data has become an important factor in defining how to process and transport the data. While looking at secured communication between the different systems in general, the need to process the data and execute code on devices created a new issue. As we need to keep code away from other users and applications has created a problem of interest to many companies and researchers. Because the creation of a program is expensive and time taking. So, in recent days, much research on ways for providing secure execution of program code on untrusted devices is developed. There are many ways to achieve the goal of data and code confidentiality on remote devices. TEEs are one among them to provide such execution. And Intel SGX is one example of the TEE which provides this need.

This work aims to give good awareness about recent technology introduced in the Intel processor, with Skylake architecture under the name Software Guard Extensions. The base of this work is to provide security features of Intel SGX by which the application’s data and code are protected from the underlying hardware and vulnerabilities. So even if the OS is corrupted the application is safe. But it has limitations because of its vulnerabilities. This thesis describes the possible attacks exploited by researchers on Intel SGX as it has vulnerabilities. Also, there are no real-time cyber-attacks on SGX were recorded this work suggests some mitigations and protection mechanisms so that in the future no real-time attacks or breaches should not happen when using SGX.

1.4 Related Work

SGX is a technology started in 2015 for 6th Gen Intel microprocessors based on Skylake microarchitecture. There is a limited amount of research available about this technology as very less applications make use of this technology. Jaak Randmets has explained an excellent write-up on a full analysis of mitigations and vulnerabilities of SGX Apps along with a comprehensive description of attack vectors and countermeasures.

As SGX claims that it is quite strong security guarantees. A lot of technical details on attacks on SGX has detailed in my thesis. I am not trying to attack but discuss possible security issues, mitigations, and the hardening methods for Intel SGX. Zhang [37], Oleksenko [19], and Townley, Daniel [34] did defense against cache

SCAs. There are some notable papers [32,26] that describe SGX applications. These papers are concerned with the secure use of SGX, and we discuss some of their solutions in chapters 3 and 5.

Another concept that provides security features to a device is TPM (Trusted Platform Module). It is a hardware module, and it requires generating symmetric and asymmetric encryption and may create cryptographic hashes. It has some data storage capacity and offers cryptographic functions for this storage to perform integrity checks on the application or platforms using it. This cannot be used for the general computation it's just used only as a cryptographic co-processor.

One more possibility for trusted execution is smart cards [66], which are small chips that secure the physical environment. If someone tries to get control of the physical protection of the chip this comes into place. It has a little connection interface and can be accessed remotely also. One example of this smart card is a credit card. Its connection interface is standardized, and it does not process its power source on its own. In general, a smart card is a little computer and is powered when it is connected to a device. It can be authenticated when it is connected to a device and remote users. After authentication computation is done on the smart card. It uses cryptographic functions so all the data entering and leaving the smart card is secured. These are called cryptographic smart cards. There is one more type of smart card called the java smart card for general computational use.

1.5 Literature Review

In this subsection am going to detail the way of finding the research publications including libraries, along with keywords and selection basis.

Search Repositories: To get wide coverage, collected SGX- applications, and vulnerabilities-related research papers from famous libraries like SpringerLink, Google Scholar, ACM Digital Library, the latest Intel documentation, and IEEE/IET Electronic Library.

Keywords: Used “Intel SGX”, “Attacks on SGX”, and “Applications of SGX and TEEs” as keywords for the subject of research. Any paper that has mentioned any of these keywords in the abstract, title, and keywords is taken for the research.

Selection: This paper is an overview of Intel SGX's advantages, limitations, attacks, and mitigations of side-channel attacks. So SGX termed papers belonging to this basis are summarized. Around 86 references are analyzed to describe the whole thesis topic.

1.6 Main Contributions

- In Chapter 2 Firstly introduces the secure enclave, enclave memory organization, enclave calls, and enclave debug mode. Secondly about other TEEs hardware technologies and a comparison between Intel SGX vs ARM SEV.

- In Chapter 3 Introduction to SGX, enabling SGX, an analysis of Intel SGX, its general features, and security features like data sealing and attestation.

- In Chapter 4 Advantages and Limitations of Intel SGX. And vulnerabilities and protection from side-channel attacks. Some Randomization and Obfuscation techniques.

- In Chapter 5 Intel SGX for Client-side and Server-Side. Mainly client-side attacks of the password manager and Server- Side applications of the Intel SGX.

CHAPTER 2

2. Introduction to Secure Enclave

Trusted Execution Environments (enclaves/secure enclaves) is a safe area of the processor which protects the application’s execution, data, and code even if any malware is present in the system [1].

TEEs are software environments that provide features for creating and running programs as it involves code and data and it secures them concerning confidentiality and integrity properties. Enclave secrets [Figure 1] are only accessed by the code inside it. And this code inside the enclave will not be altered by unauthorized users. (chapter 3) Intel SGX enables the creation of a protected memory region known as an enclave [14]. This runs on the application's virtual address.

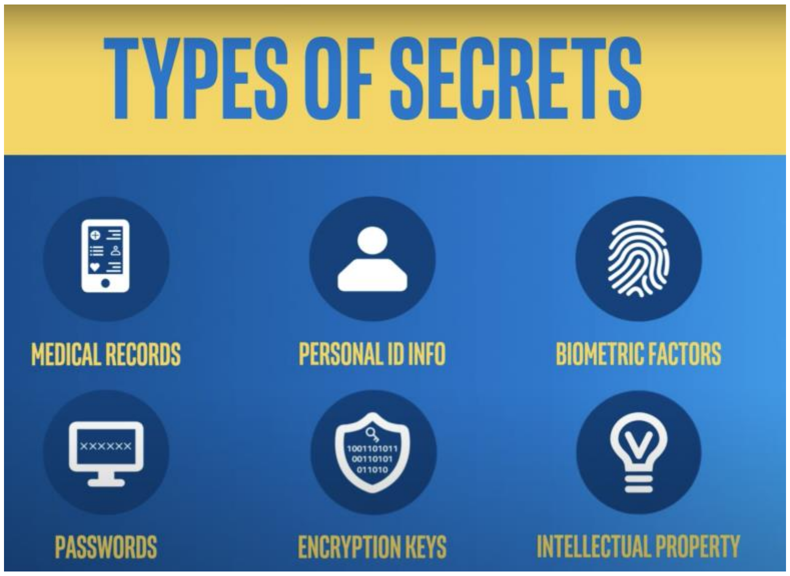

Figure 1: Enclave Secrets [76]

Enclaves have no state: They are terminated in the following situations firstly while the system is in sleep mode, while the application is turned off [16], and also whenever the application explicitly stops them. More importantly, when an enclave is terminated all its content is gone.

Nowadays TEE is widely used in systems for boosting the security of software execution. SGX enables an application to run code and secures sensitive data in its TEE.

In simple terms, Enclave is in the DRAM memory space. For example, DRAM space is 4GB out of this 128 MB is enclave so from address number 0 to address number 128 is the enclave address. So, the enclave page cache will start mapping to any virtual address which is part of this 128 MB in this DRAM [23].

TEE is a computing environment that facilitates the creation or running of secure code or processes. This TEE is called a secure enclave as per Intel SGX. In general terms, TEE offers a greater level of security for applications executing on a system than an application running on a rich operating system.

The basic protection provided by SGX is enclave secrets are only be used by the code which is inside the enclave. The only method to execute this code is by using the interface functions of the enclave, which developers have implemented. CPU does this restriction.

Every enclave describes an enclave call called ECALL which is the entry point into the enclave from an untrusted application. The enclave may have a call from outside called OCALL that permits enclave functions to call the untrusted application and then goes to the enclave [24]. Collectively these calls are called an enclave interface, but these functions are not directly executed by our program. Because access to enclave place entry and exit points is tightly controlled by the processor and special CPU instructions are required to make these transactions.

The Intel SGX software developers kit abstracts these low-level details to provide software developers with a familiar programming environment.
A special tool called EDGER8R will automatically generate proxy functions named after ECALLS and OCALLS. So that application can invoke them as it would any other C function.

To create that proxy function tools read the enclave interface definition from the EDL file. An EDL file looks like a C language header file with a function prototype. The EDL file contains two sections Enclave Trusted and Untrusted Functions in

EDL (Enclave Definition Language). ECALLS are defined under the trusted function and OCALLS are defined under the untrusted function.

```
enclave {

trusted {

/*here write ECALLS*/
Public first_ecall (int value);
 int ecall_private (int value);

};
untrusted {
/*here write OCALLS*/

};

};
```

These above functions are prototyped much like the C programming language but some additional key values are needed. All ECALLS that are meant to be invoked by the untrusted application must be declared as public. Every enclave must have at least one public ECALL. The public keyword precedes your function name. An ECALL without public designation within another OCALL.

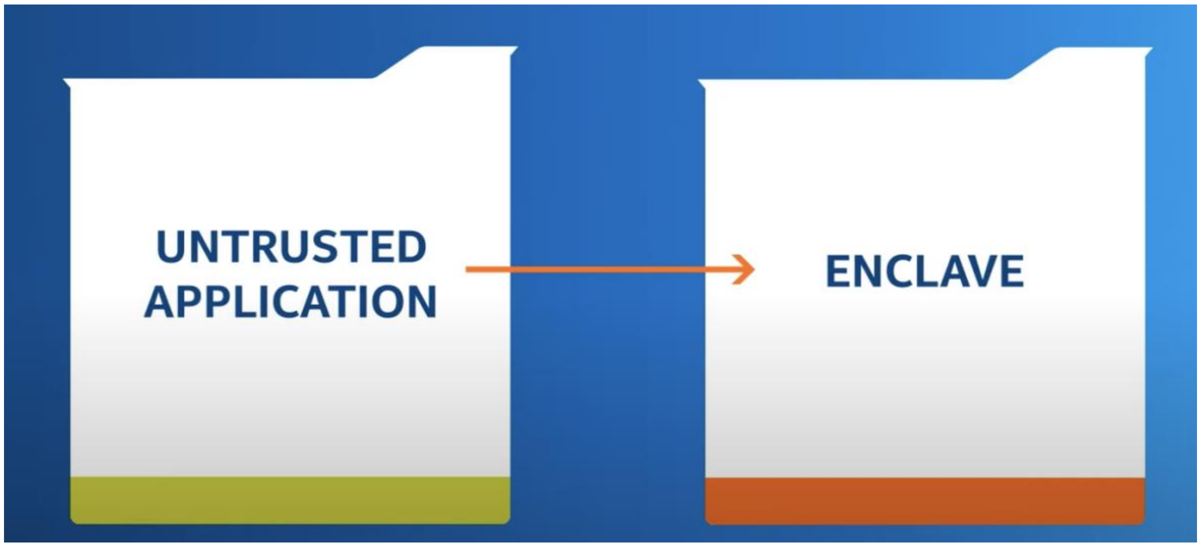

Figure 2 ECALL [77]

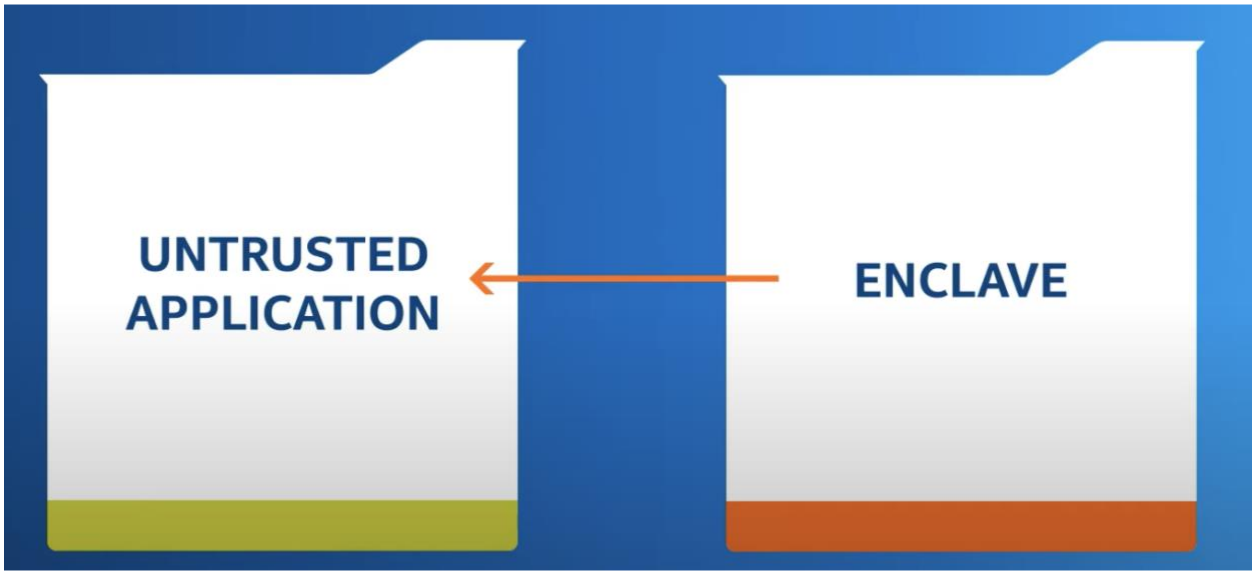

Figure 3 OCALL [77]

When ECALL or OCALL is made the function parameters are marshaled between untrusted memory and trusted memory. When a parameter is passed by a value this marshaling is one way. Changes made in ECALL or OCALL are not propagated back to the calling function. When a parameter is passed by reference (int32 *value) you must completely describe the data marshaling each point or parameter is preceded by ([] int32 *value) that describes the way of the data marshaling and the number of items to be marshaled. The ([in] int32 *value) in a keyword that the data should be marshaled in ECALL or OCALL. The ([out] int32 *value) out the keyword that the data should be marshaled out ECALL or OCALL back to the calling function.

When passing data out the data buffer must be allocated before the ECALL or OCALL is made, and it must be large enough to hold the data. Specifying both ([in, out] int32 *value) in and out data in both directions. By default, the edger8r utility assumes that the data buffer consists of one element with the size of the argument type.

In the example code, one 32-bit integer will be marshaled.

In Figure 4 you can see EPC which is called an Enclave Page Cache which is achieved by splitting 4KB pages into different enclaves.

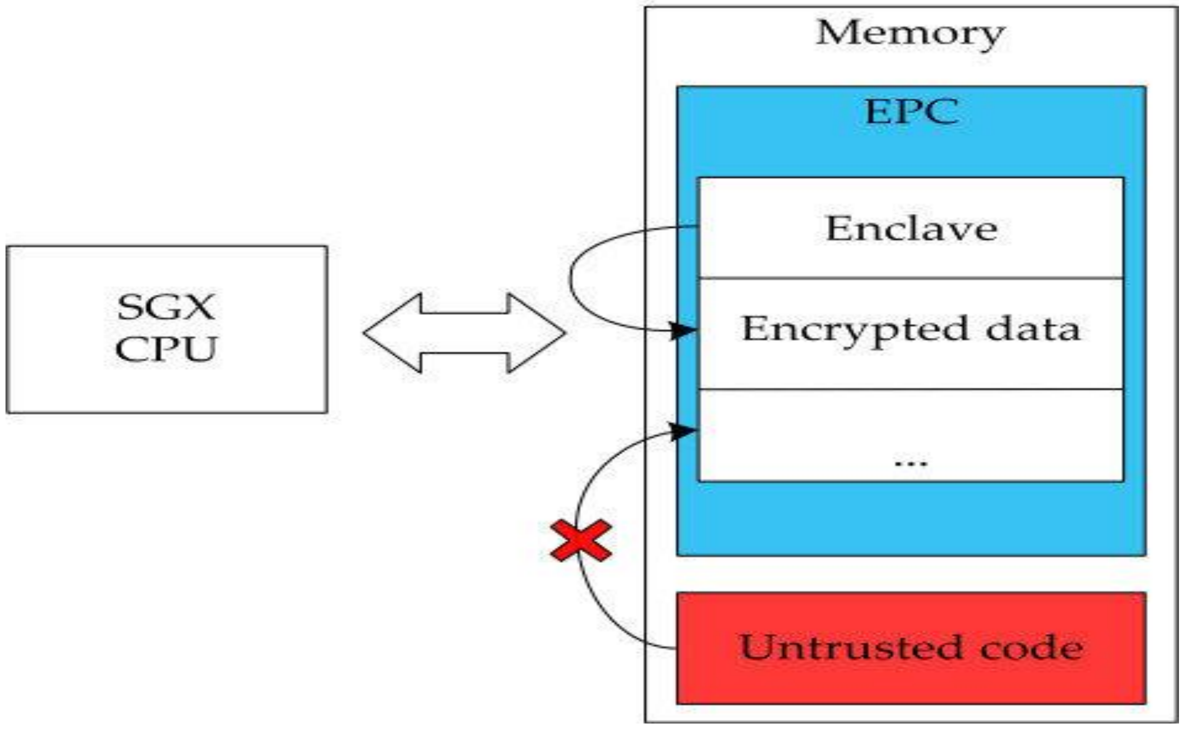

Figure 4 Intel SGX and Secure Enclave [81]

2.1 Memory Organization of Enclave

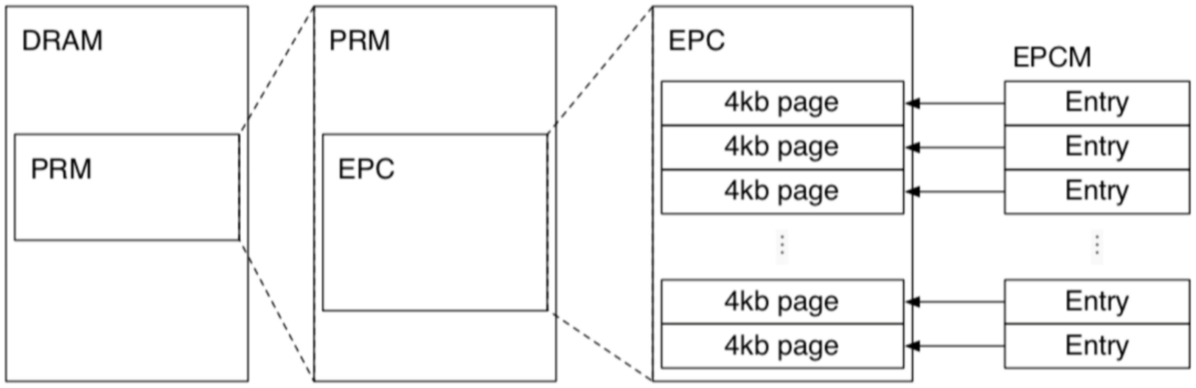

Figure 5 Memory Organization of Enclave [80]

DRAM:

It provides memory for software written for Intel architecture with the input-output devices via the input-output address space which is primarily used for access to the DRAM.

Accessing DRAM is untrusted every time writing something into it like the last level cache will write back requests on a replacement from the LLC. It will go through an encryption engine. And similarly, if sending a load request that is missing in the cache hierarchy and coming from DRAM then we have to decrypt it. This encryption and decryption will add an extra layer of latency on top of existing memory hierarchy latencies. Because of this, the performance will go down.

PRM(Processor Reserved Memory):

Data and the Code of the enclave are stored in PRM. This is the under the DRAM which will not be accessed by other applications even by the system software. Direct Memory Access is rejected by the CPU to protect the enclave from other peripherals.

EPC(Enclave Page Cache):

Enclave data is stored in this area of memory which is only used by the CPU**.** When an enclave data leave the CPU, it is encrypted, and hashed by protection using a key that can be accessible only to the CPU. This is an advantage that enclave data is thereby secured from privileged software-hardware attacks.

EPCM:

As it is necessary to perform security checks as SGX records a small amount of system software allocation for the EPC page in EPCM.

2.2 Hardware Technologies

Hardware-assisted TEE is secure isolation technology that has served as an efficient defense mechanism for providing security at the system level [4].

TEE is not just limited to Intel® SGX but there are many hardware technologies [2] that support TEE implementation such as AMD Platform Security Processor and AMD secure Encrypted virtualization, ARM TrustZone, TPM, x86 System Management Mode, and Intel Management Engine.

2.2.1 ARM Trustzone

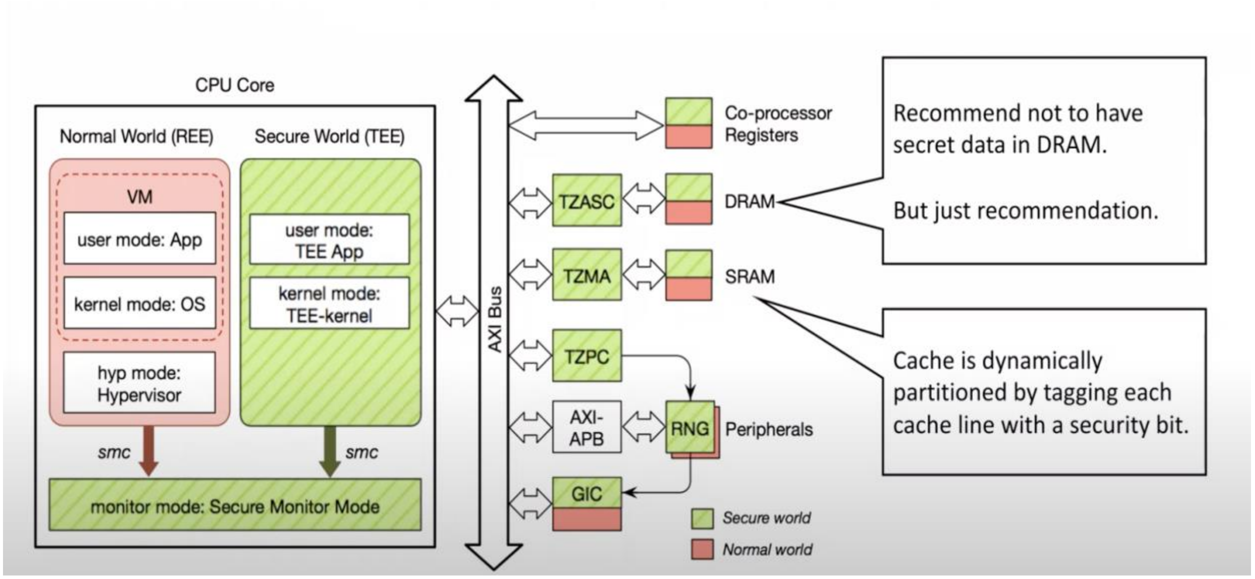

Figure 6 ARM Trustzone [84]

For mobile phones and all the handle devices, ARM processors are there which are like Trusted Execution Environments. But here there is no notion of data storing in encrypted form. And have a security monitor with two modes: normal world and secured world [25]. The moment we want to execute security-critical code then the CPU can jump into secured mode. When we compare ARM with Intel® SGX, ARM is designed in a more disciplined way like from registers, cache (SRAM), and DRAM are partitioned properly. There is no possibility of side channels. ARM is very hard to attack when compared to Intel® SGX. But the downside of ARM is the complexity of the development process and mostly no device can be fully hackproof. ARM TrustZone technology acquires a process of splitting system resources between 2 execution worlds, the real world, which is unsecured, and the secured [3]. This TrustZone is programmed within the hardware by enabling the protection of peripherals and memory. Also, it does not care about the vulnerabilities which are caused by proprietary. This is how the performance of the security is continued. This mechanism is practiced for mobile payments and other such applications on the secure kernel [32].

The trusted part of the Intel SGX is the CPU similarly for ARM TrustZone the trusted part is firmware and hardware. But there is no protection against hardware attacks in the ARM TrustZone, but we can protect against hardware in Intel SGX. In general, there are no trusted peripherals in Intel SGX, but ARM has it [64].

2.2.2 Intel SGX vs AMD SEV

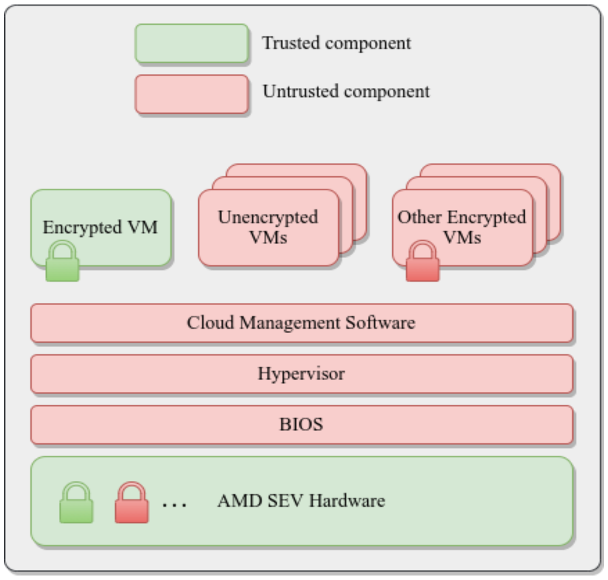

Figure 7 AMD SEV [85]

AMD has a similar trusted execution environment to SGX, but this is more scalable when compared to Intel SGX. Like when SGX arrived on the market it has a memory size limited to 128 MB for one enclave space or EPC. But later it jumped to 256 MB. But AMD does not put any limits.

Intel counterpart does attestation by generating a key, but AMD has put a secure co-processor inside the chip. So anytime you access using AMD and verify yourself like whether you are supposed to access the address or not, you can send the key to a secure coprocessor.

By looking at the page size limit one can understand that Intel SGX was designed for executing the small and highly secured code. But AMD can run large enterprise apps inside enclaves. Like a whole big, DB can be run inside an enclave for AMD technology, where SGX gives you page faults because only 128 MB is allocated to an enclave.

Table 1: Comparison between ADM vs Intel SGX

S. No	Hardware	Main Technology	Explanation	Memory size (Limits)	Access Level (Highest)
1	AMD SEV	SEV SME Virtualization	This safeguards virtual hosts by encrypting physical memory data and then protects them from malicious cloud service providers.	Until available system RAM	Ring 0
2	Intel SGX	Enclave EPC	This envelopes an application's security functionalities for an enclave and prevents malware.	Up to 128mb of EPC	Ring 3

2.3 Enclave in Debug Mode

SGX's enclave can develop in debug mode or release mode [14]. Debug mode of an enclave is scanned and will be able to attach it to SGX's debugger and inspect its state. Can even also enter from its code like executing other applications. When actively developing an Intel SGX application then that is called debug mode of the enclave.

The CPU permits the debug mode of an enclave to launch but it may attach to the debugger if it is not secure. Enclaves default build in debug mode must not be deployed in real-time applications. Unlike the debug mode an enclave built-in release mode can't be executed in any situations. This is done by the processor.

Debugging in the SGXs enclave does require the utilization of the particular CPU instructions and this means having to utilize the SGX debugger which is enabled in the SGX application development kit. People who debug without using SGX simply avoids enclave code.

2.4 Two Use cases of TEEs

This subsection describes two use cases of TEEs they are DRM and Malware prevention and Detection.

- **Digital Rights Management**

TEE can be applied as DRM on user devices by restricting unauthorized access to copyright content, like movies, applications, songs., etc. Any company can distribute such content with the approach to DRM using the TEE by avoiding legal concerns. So first they distribute the encrypted content like an encrypted video song for loading in advance and next download the song, yet it is first it will not be allowed to play by the user. But if the user wants to see the video song his system should initially access the licensing server for getting a key and then the server validates the payment or license and provides an access key. Here comes the TEE to decrypt the video song with the key provided by the server and then the user can watch the song. The original copy of the video song is saved in the TEE and then the purchaser can't take out the original copy of the video song to share with other people It authorizes a distinct monetization strategy. Let's say the user can only watch the song for up to one month. Post this month finishes, TEE blocks the purchaser's access. Moreover, TEEs may be used to protect in case of stealing copyrighted content, like recording the video using screen recorder applications and distributing illegally. An example of DRM is Microsoft's Play. DRM can be implemented using Intel SGX

- **Malware Prevention and Detection**

  Malware attacks are increasing these days which means user content is at threat, and general methods like static or behavior malware analysis are not enough. Mainly malware (like ransomware) encrypts the user's contents and utilizes this to earn revenue is increasing. This may have more effect on the users. There is a way to use TEE to store important information in the protected storage of enclaves. This helps to be away from malware manipulating the data. One way to implement for detection of malware is by Samsung Knox. This is done by a feature known as Knox Attestation [69], here TEE can be used to verify the status of the device if it has tampered with any malware. In the case of malware prevention and detection, SGX may be utilized as it depends on the secure area to confirm the health of the system software.

Along with these two applications TEEs can be used for other purposes like Mobile identification, Mobile transactions, Anonymous attestation, and many other purposes.

CHAPTER 3

3. Introduction to SGX

Intel® SGX is a great example of a TEE [22]. SGX uses enclaves to isolate execution environments from other applications. Intel SGX isolates secure code and data creating an enclave that is at the application level. It is the set of extensions that aim to allow confidentiality and integrity of data-in-use for secure computations in systems [27]. If we understand the notion of OS rings, it's at the ring 3 level.

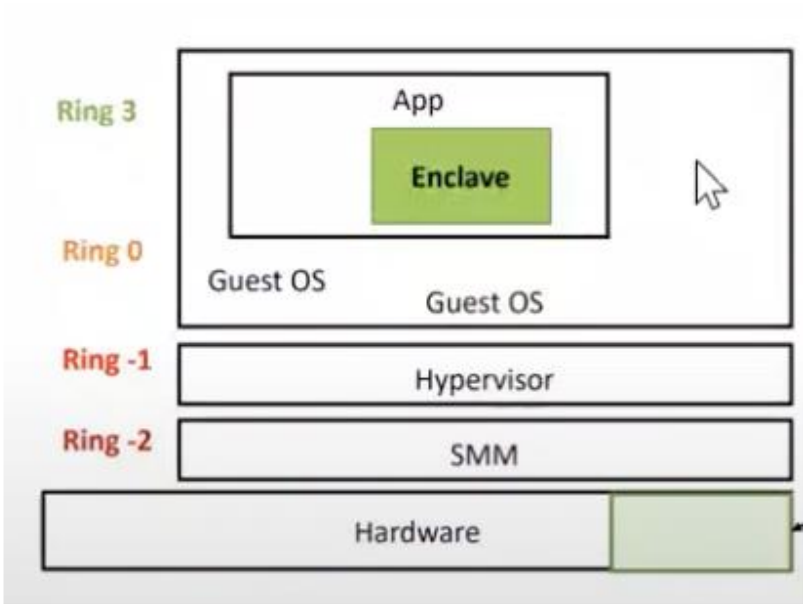

Figure 8 Isolation [79]

3.1 Intel Software Guard Extensions

In late 2015 Intel made a processor with SGX [4] [18]. Intel® SGX is a hardware trust mechanism and hardware-based memory encryption which separates particular applications code and its data inside the memory.

It allows an application to execute code and protect sensitive data inside its enclave, by allowing developers to control directly for application security. It protects code and data from outside software environments.

SGX is built to secure the application's confidential information from nasty software. SGX guarantees confidentiality and integrity for a security-sensitive execution performed by a system by a set of extensions to an Intel architecture where all the other software (OS, hypervisor) on the system is potentially insecure [3].

From Figure 10 we have an OS USER process where we have a code portion and data portion, so if we want something to be secured like the code and data part to be secured like need confidentiality and integrity there comes SGX i.e., [12] series of instructions provided by Intel® by creating a memory area or space inside DRAM where particular space is allocated to a particular enclave or the secure process will have its code and data. Enclave has restricted memory pages.

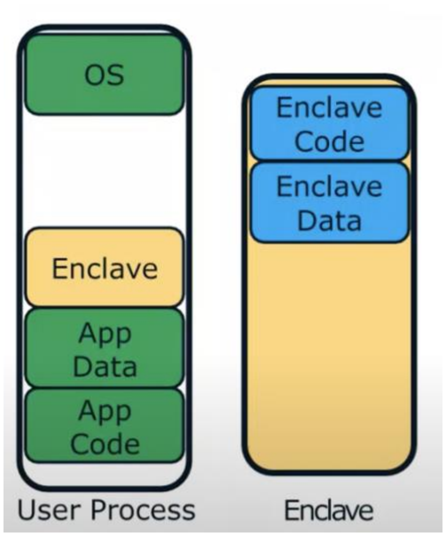

Figure 9 User Process and Enclave [82]

In Intel® SGX there exists TCB, a very small one when compared to other systems, which is known as the set of computing technologies that should work correctly and must not be malicious for a security system to operate. If TCB is larger then it's easier for something to be wrong. So, it will be small.

SGX is the technology implemented for the Intel Core processors. Intel calls SGX a reverse sandbox and not a sandbox because the developed code and its data [29] are obscured from the entire system. Instead of isolating untrusted applications or malicious software, it protects the developed code. This technology was firstly intended for the protection of source code and data of the application developers from unauthorized alterations and access. The main point of this technology is to create a software enclave that is a protected memory area and access to what is controlled by the processor will only be allowed to the user application that created the enclave.

SGX is greatly employed to improve the safety of an application, yet it’s very essential to point out human aspects and organizational perception of safety. Because sometimes data might be leaked because of human error even without attackers involved, for example, lack of knowledge, or coding errors. Moreover, there is no mitigation when the platform is not up to date.

SGX depends on 3 mechanisms to protect data they are sealing of data, attestation (chapter 3.1.3), and enclaves [31] by isolation.

3.1.1 Enabling of SGX

SGX allowed processor provides us with a trusted computation by isolating an enclave's environment from suspicious software which is outer to the enclave [3]. Application installers must detect whether Intel SGX is supported by CPU and BIOS.

Earlier to an application can utilize SGX 4 below things should be done

1. Firstly, the processor of that device should assist SGX instructions.

2. The Basic Input/Output system should also support the SGX.

3. Then enable SGX from the BIOS

4. Finally, SGX platform software is installed on that device.

   SGX was introduced with sixth Gen Intel core processors and the Intel Xeon E-3 v-6 server processor.

3.1.2 Features of Intel SGX

- Nonaddressable memory pages are booked from the system's physical memory and then encrypted.

- It has an architectural property that introduced a modern set of CPU instructions allowing the app user to build and utilize the hardware-based TEE which is known to be an enclave [4].

- It mainly assures the Integrity & Confidentiality of the enclave's data and code during the runtime even if the OS or hypervisor is compromised.

- It is the only hardware that provides memory integrity protection.

- It separates carefully the untrusted and trusted environments and provides a protected and narrow enclave gateway.

- Can perform attestation, verification, and everything even locally or can-do remote platform like send to a remote server which will verify and send you a green signal.

Table 2: Security Features of Intel SGX

S. No	Security Feature	Definition
1	Encryption (Secure Storage)	Data stored in the DRAM is encrypted, a secure and privileged memory area.
2	Data Sealing	It helps to protect enclave data at rest.
3	Attestation	Helps to evaluate enclave identity. And remote attestation enables third parties to authenticate the integrity of the TEE and establishes confidence between both parties.
4	Isolation (Isolated Execution)	SGX isolates the normal world and secure world therefore by preventing malicious apps and isolates sensitive data and code.

3.2 Attestation

Before accessing anything, you are verifying yourself and let's say the verifying entity somewhere else in the world, so we send a request that you are worth accessing the machine. Initially, the key is generated by the CPU as it only trusts the CPU. This process is only for verification purposes. The moment the creation of the enclave happens at that time attestation is required as we send the key to the attestation entity and post verification, we get access. Let's say we have a web application and want some part of it to be secured like SGX-related code which will be part of the enclave then only you need to verify everything is correct. So, at this stage, we use Intel® SGX run time environment and put the code inside the SGX environment. Then it will initialize and verifies that these are the address space that you are supposed to use and no one else could be supposed to access is attestation.

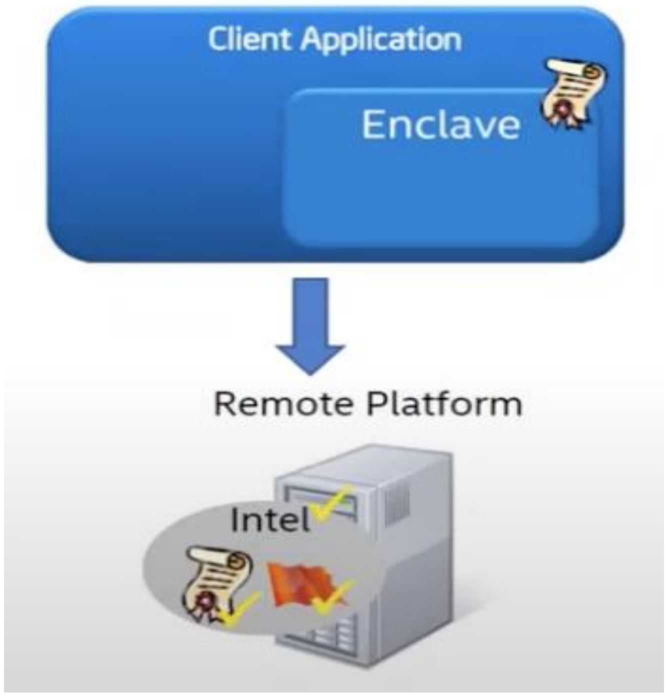

Figure 10 Remote Attestation [79]

SXG helps with two attestation strategies. The first one is local attestation which is used to build a connection between 2 distinct enclaves on one system and for ensuring integrity, confidentiality, and reply protection. Another strategy is remote attestation which is utilized to guarantee intactness and integrity of the hardware when an enclave to a third party is not placed on the same device. This is done to gain the trust.

Local attestation comes into the picture when two enclaves on one system want to work jointly. For this purpose, enclaves need a safe passage by which those two enclaves can interact. For this strategy, SGX uses a secure algorithm called Diffie Hellman Key Exchange and can be utilized by enclaves to share a symmetric key by an untrusted passage

Table 3:SGX Instructions

S.NO	Execution Mode	Instruction	Description
1	Ring 0	EADD	To add an EPC page to the present enclave. It is utilized for loading data and code.
2	Ring 0	EEXTEND	This instruction used when updating the enclave measurement while attestation. That is modifying the SECS.
3	Ring 0	ECREATE	This instruction is used to copy and generate the SECS structure to EPC's new page, that is for initializing a new enclave.
4	Ring 0	EREMOVE	For permanently removing a page from EPC. Generally, while enclave destruction.
5	Ring 0	EINIT	For enclave initialization and finalization for its measurements and attributes.

For remote attestation, SGX has different components in its processors, that have special enclaves and one-time writeable memory. An important component for remote attestation is the measurement of an enclave. It is utilized to find the execution of the software inside, whereas the third part compares the measurement with the expected value to check the code executed inside an enclave. This happens whenever a third party wanted. The measurement is computed with the help of robust hash functions. This hash value is the input to the EADD, EEXTEND, and ECREATE instructions.

Enclave is outlined by its SECS – SGX Enclave Control Structure. This is generated whenever an enclave is created and then saved in a separate entry in the enclave page cache. SECS carries important data which includes the enclave's global identifier, its memory usage, and measure hash of its (MRENCLAVE section 3.3.1) Table 3 instructions are used in the enclave lifecycle.

The execution mode is Ring Zero is the most privileged ring (Figure 12). The OS often runs in this ring, it has all the privileges of the hardware resources. All the applications in Intel SGX are executed at Ring 3 and this ring is isolated from the Ring 0 i., the system software. This isolation protects upper rings from malicious code or faults from the lower rings.

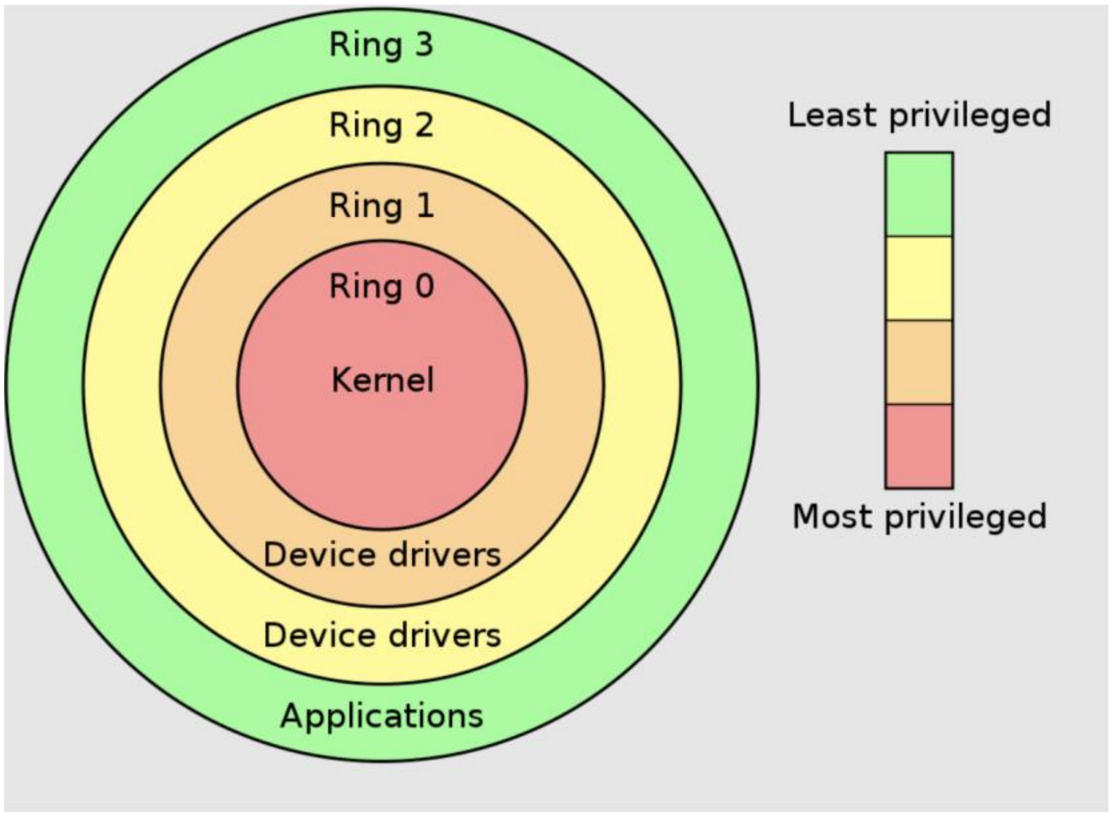

Figure 11 Privileges of the architecture [67]

SGAxe is a transient execution attack, not a real-time attack, it is used to recover SGX attestation keys. This execution breaks the great feature of SGX and shows SGXs weak ecosystem [74] and this tells Intel SGX fails in practice. Seeing this intel released new patches and updates but still new transient execution practical vulnerabilities are released which means we can’t have control and discover the emerging transient execution vulnerabilities [75] in reality.

3.3 Seal Data in Intel SGX

During some situations where Intel® SGX applications may need to save secrets outside of its enclaves and in **chapter 2** Introduction to enclaves, you have seen that enclave content is lost in a few situations. So, save the content which is stored inside an enclave should be explicitly sent outside the enclave to untrusted memory. Seal and unsealing of the data are done through a cryptographic library to protect data outside the enclave [16].

While accessing the untrusted memory the data stored is encrypted using a key provided by the CPU. At a higher level in the cryptography part in Intel® SGX, it assumes the CPU is secured but optimal memory is not secured. The problem you know about untrusted memory is that we can't trust it. It can be looked over and can even be altered by a malicious user. So, it makes sense that data stored in DRAM is encrypted. If we must get the data from DRAM, we must decrypt it and there will be additional latency because you need to perform some additional checks along with normal load requests.

To solve this issue SGX provides a way called data sealing [14]. Data is sealed means it is encrypted inside the enclave by an encryption key that is generated from the CPU. “This encrypted data block is known as sealed data” which can only be unsealed on the system where it is created. Encryption provides a guarantee of integrity, confidentiality, and authenticity of the data.

While sealing your data, you can choose from one of two key policies MRENCLAVE and MRSIGNER. These two policies help the generation of the encryption key. These two policies are also used for the attestation process.

Page abort is used when non-enclave access or non-enclave code/data is trying to access which is in the enclave.

3.3.1 MRENCLAVE and MRSIGNER

MRENCLAVE policy will derive a key that is specific to that enclave on the same system which means only that enclave can decrypt the data on that system. MRENCLAVE is a unique identifier for the contents of the enclave. MRSIGNER policy will derive a key that is specific to developers signing the key on that system. This will allow the sealing of data by an enclave to decrypt by another enclave by that software vendor on that system. We can also share data between two different enclaves in two different applications.

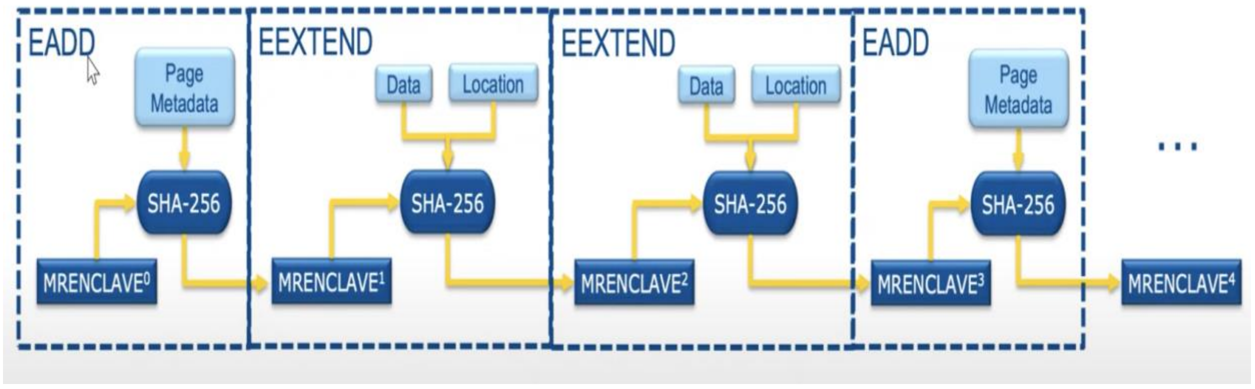

Figure 12 MRENCLAVE and The Key [79]

MRENCLAVE is generated with the help of hashing the instructions. Then data is passed when the enclave is created using the instructions EEXTEND, EADD, and ECREATE. Finally, the value is stored and finalized in SECS of EINIT. **Figure 11** Is an example: let's say EADD is an instruction for adding an enclave. EEXTEND is to extend the memory space that you have allocated for the enclave so every time you extend it you generate a hash. As for my example, let's say this is the entire sequence that you have generated while creating or using an enclave. MRENCLAVE4 is the final hash for your enclave.

This final hash is the key to the lock to access anything(data/code) that is supposed to access with the final key which is stored in the different parts of the memory. This is the access control mechanism to put it at a very high level.

In both of these key policies, the computer that is sealing data is one of the key derivation inputs. If the sealed data is copied to another system that system will not be able to decrypt the data even if it is running the same application [18].

Another input to key derivation is the debug mode of the enclave. Enclaves that are built in debug mode can't decrypt data that is encrypted by enclave built-in release mode and vice-versa. It is the security mechanism to prevent the Intel® SGX debugger from using a debug enclave to expose secrets that were sealed by the production Intel® SGX applications.

There is an important restriction that needs to be remembered while using Intel® SGX is data sealing which might have important security issues. Enclaves don't verify untrusted applications. One should not believe that only your application will load your enclaves i.e., anyone can load your enclave. Your enclave APIs should not allow for the sealing or unsealing of data capabilities to leak secrets or allow unauthorized access.

Data sealing is an important capability offered by Intel® SGX. With the help of it, you can safely export your data outside an enclave and be sure that secrets have not been manipulated, alerted, inspected, or even copied to another computer system.

3.3.2 Access Flow

The notion of linear address is nothing but the virtual address. Then next page table checks will happen and generate physical addresses. After these additional checks will happen firstly will it check enclave access or not? If yes, then it will check if the address is in the enclave cache or not. And then it will check the enclave page cache manager or not.

If all the checks have passed, then it will allow memory access, or else it will generate a fault. If the request is not enclaved access but the address is with the EPC that means a non-enclave request is trying to access something which is in the enclave region so it will generate a page abort. And no memory access happens when the page abort. And if the address is not in the enclave cache that means it's a normal request to a normal OS page so it will have memory access.

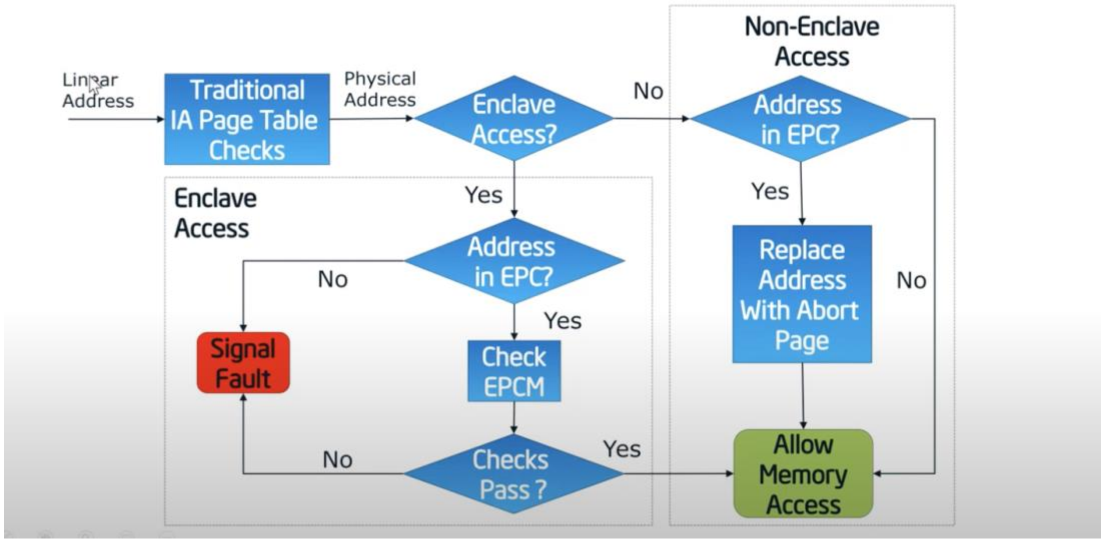

Figure 13 Access Flow [79]

From the architectural point of view, this is key. Some of the recent attacks on Intel SGX or some of the early attacks on Intel®SGX exploited this part. And Intel® SGX doesn't take care of cache. Every time we see some attacks on the cache. And the first attack that came on Intel SGX is page faults.

```
Void inc_secret(void)
{
  if (Secret)
   *c += 1;
 else
   *d += 1;
}
```

<u>Page Table</u>

PTE c

PTE d

Let's consider the below program to show an attack on this part like at a very high-level page table-based side channel, especially with enclave is by looking into below simple function:

Where *c and *d are two different memory addresses i.e., pointers and memory addresses mean different OS pages. And data flow and control flow dependency here according to the outcome of the value in secret, you access a particular page. About page abort from the access flow if someone is not inside the enclave trying to access something which is in the enclave i.e., page access flow even if it generates a page abort it goes ahead and then tries to get the page table entry and it caches the page table entry. So, after the dataflow and control flow dependency, the cache hierarchy will have the page table entries, even if we abort it. As an attacker, someone can send a request to enclave addresses which will eventually be aborted as an outsider who tries for enclave access, but the page table walker will actually go and then get the page table entries from DRAM into the cache hierarchy if it is not there in the page table.

3.4 Analysis of Intel SGX

An attacker cannot read any data of the enclave as it is encrypted while it is in the device's memory. Moreover, the attacker can still modify any part of the encrypted enclave as the enclave is verified when it is utilized which means the attack will be noticed. Enclave will be initialized again to equate to the real content that the developer intended. Therefore, the developer need not trust the entire device that the program runs on and only trust the CPU that the code is executed on. SGX allows separation from different processes executed inside enclaves, which means the execution of one enclave does not impact any other enclave.

There is a disadvantage of using Intel SGX is the enclave needs to be decrypted every time it is executed. And post-execution the entire enclave must be encrypted before it is stored in the device memory. These two operations increase the application execution time. Finally, to sum up, Intel SGX gives an environment to execute code without the involvement of the other processes and provides integrity and confidentiality for both code and data. Moreover, SGX allows remote verification for the created enclave. Lastly, one more disadvantage of SGX is it can’t be protected against side-channel attacks which means some of the information about the enclave can be inferred.

CHAPTER 4

4. Advantages and Limitations of TEE

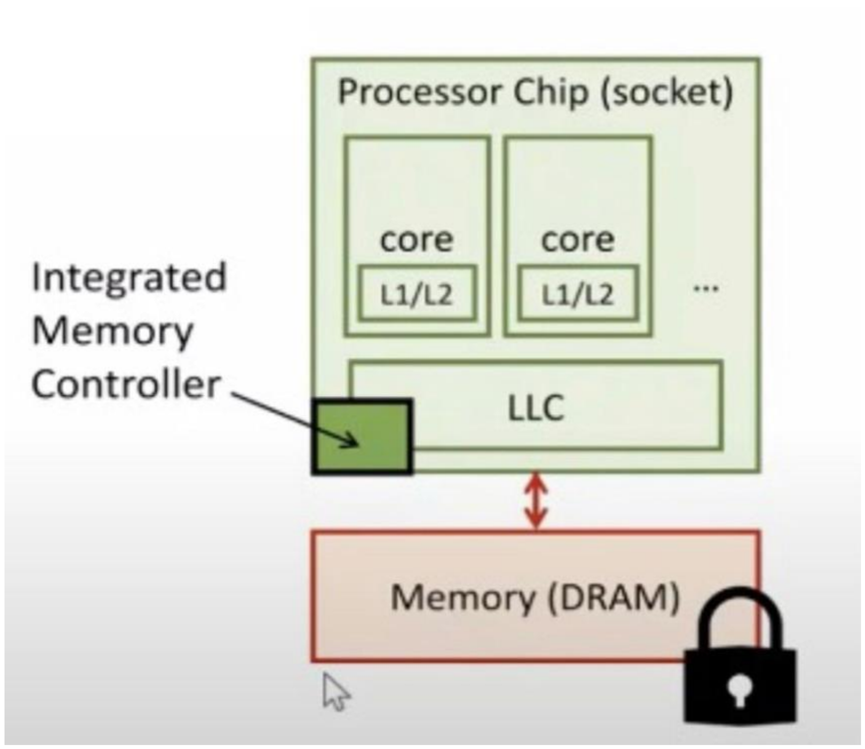

Figure 14 DRAM Protection [79]

At a high level, we can assume that the CPU is secured but option memory is not secured as memory chips are coming from different third-party vendors, so it assumes the data present in DRAM is encrypted. So, DRAM is considered to be untrusted and that's the reason the data is stored in the encrypted format. Every time you access DRAM you have to decrypt it before you get the actual data.

4.1 Advantages of TEE

- The main advantage of TEE is hardware isolation i.e., the general world and secured world, and memory protection (in Figure 2). Therefore, prevents malicious apps and isolates important data [6], and enables code running in the protected world to additionally secure against software attacks [7].

- Other important advantages of TEE are low-performance overhead and low overhead on power consumption [8].

- Low overhead on latency [9].

- Resist against physical memory access attacks [4].

- A lot of real-life applications take the benefit of the hardware-based and enforced integrity and confidentiality guarantee which is provided by Intel SGX.

4.2 Limitations of TEE

- **TEEs** is lacks protection from side-channel attacks [10] like power analysis.

- There is an incoherence between the cache insecure world and the secure world which is related to the TrustZone as someone may try to exploit it as a weakness.

- The use of particular hardware is also considered a disadvantage [11].

- As TEE provides secure execution but if code running in that is buggy then TEE is untrustworthy [2].

- As per TCB anything that is not a part of the CPU is untrusted similar to OS.

- SGX is not implemented to handle reverse engineering or side-channel attacks. This is up to the developers of the SGX to code enclaves that are secure against such kinds of attacks.

4.2.1 Vulnerabilities of SGX

Some Vulnerabilities of SGX are Side channel attacks, Software Vulnerabilities, and Brute Forcing for Decryption by attackers [4].

Side channels:

A side channel is a channel that is not for information transfer, however information leaks due to the system's physical construction [72]. The security of TEEs such as Intel SGX has been notably threaded by side-channel attacks. The problem of writing an algorithm with the purpose that it is side-channel secure is important. Because if the bad user can have physical access to the system and then the device should ensure that nothing can be leaked by calculation of performance like from power. It is very hard because different side channels are continuously identified [26] by the researchers.

We can see from 4.2.2 that it protects from a few attacks, but it can't escape from side-channel attacks which can collect figures from the CPU about execution and possibly deduce the characteristics of the software execution. Some examples include power statistics, performance, cache misses [13], and cache timing attacks [17].

Software Vulnerabilities:

SGX components are complicated, so it is likely to have bugs like other software. Enclave has a small memory space, and this could lead to a brute force mechanism so it might help to find the right address. The probability of success of attack increases with the injection of NOP slides before malicious code. This is because SGX is built to protect against normal attacks, but practical work studies show that SGX should concentrate on the software vulnerabilities for the trusted instructions [73].

Decrypting:

To get important data from the enclaves bad actors may try to intercept the secure channel to obtain the remote attestation process (This process permits a remote party to look over that the intended software is safely running within an enclave in a system with Intel® SGX enabled). Enclave can verify itself by hash/signature to a remote Intel server, and the Intel server will tell that this is a valid signature for the machine that you are using

ECDH (Elliptic Curve Diffie Hellman handshake execution is done for the exchange of symmetric keys to perform remote attestation processes. So, possible vulnerabilities rely on solving the Elliptic Curve Discrete logarithm problem. For Memory encryption for encryption of enclaves and secure communication symmetric encryption is used but using certificates that are authenticated by Intel while the remote attestation process will mitigate MITM attacks.

Here in the below code for AES CTR, there is nothing that immediately seems to be weak. However, attackers try to decrypt by brute force attacks and get sensitive data but, such kinds of attacks are computationally not possible without knowing the key.

A python code for AES CTR mode

```
from Crypto.Cipher import AES #Python Script using Cryptographic
Libraries
from Crypto.Util import Counter # A counter here
import os

KEY = os.urandom(256)           # strong key for the randomness

def encrypt(plaintext): #encryption method which takes plaintext as an
argument

cipher=AES.new(KEY,AES.MODE_CTR,counter=Counter.new(128))
        #Above is AES in CTR mode with counter
        ciphertext = cipher.encrypt(plaintext) #here it encrypts the plaintext
        return ciphertext.hex()                 #returns the ciphertext
```

Transient Execution:

At the end of section 3.2, it is said that SGX is vulnerable to transient Execution attacks, and because of this attack one of the main security features of SGX called attestation is practically lost. So, what is a transient execution attack or "microarchitectural timing side-channel attack", or a speculative attack?

The latest computer devices are mostly parallel-composed and with a variety of performance characteristics. So, to explain the attack let's consider an operation that is not able to perform because the previous slow operation is still not finished. At this time a microprocessor may try to guess the solution of the previous operation and then speculatively execute the next operation, it acts as if the guess was correct. This guessing is based on the recent system behavior and when the first slower operation is completed the microprocessor tells if the guess is false or true. If the guess is true, then the execution happens without any interruption but if the guess is false then the interruption happens, and the micro process goes back to the speculative execution operation and does the original instructions along with the correct result of the first slow operation.

Generally, a speculative instruction is an instruction that happens because of an error done by a processor and affects the microarchitecture's state and for this execution, there is no trace. And you feel like there is no speculative execution happening. This affects the cache and can be only discovered by keen monitoring and observing the timing of subsequent operations. So, if any attacker can observe this execution on a system that has sensitive information, then he might be able to discover the sensitive information.

4.2.2 Ways to protect from side-channel attacks (SCAs)

SGX explicitly can not protect from SCAs. Security is mainly responsible for the enclave's developer to address the concerns of SCAs. Unfortunately, many researchers have found that the presence of side channels might leak unacceptable information from an enclave.

Adversary observes data access and timing and then designs attacks to get sensitive information from runtime computation [30]. However, mechanisms to eliminate information leaks depend on the developer to conceal access ways with other unessential accesses. Hiding ways include data oblivious execution is used. That is from the attacker's perception, this mechanism adds noise to emerging patterns from the essential computation for a naïve implementation. By using such protection methods, information leaks from an SGX enclave will reduce and will guarantee data privacy. In turn, it will add significant computational overhead to a few applications like data analytics [30] and real-time applications.

Generally, enclave operation requires an OCALL (Outside call, this call to an untrusted function from within the enclave) like thread synchronization, Input/Output, and so on may be exposed to untrusted domains. While using an OCall could allow an attacker to understand enclave secrets then there would be a security issue. This scenario is also classified as an SCA, and it is up to an ISV (Independent Software Vendor) who designs the enclave in a resilient way that will prevent the leaking of side-channel information.

If an attacker can have access to a platform and be able to see which pages are executed or accessed, mitigation of this side channel vulnerability might happen by lineup-specific code and data blocks contained completely in a single page.

An application’s enclave should use a cryptographic implementation inside the enclave which is side channel attack resistant when attacks are more concerned with side channels. Composable Cachelets (CC) is a strategy that dynamically divides the LLC (last level cache) by fully isolating enclaves from each other [34]

To prevent time-based side channel attacks Intel has designed AES-NI Advanced Encryption Standard New Instructions to be constant time. But some work demonstrated that an unprivileged might leak AES NI keys from SGX by using correlational power analysis and Linux kernel over the period of 26 hours [71].

4.2.3 Randomization and Obfuscation techniques

There are various techniques available for randomization and obfuscation to improve the security of the Intel SGX. These techniques make attacks more time-consuming as they are complex to understand and might not fully mitigate the attacks as new methods can be used to exploit them to check the vulnerabilities. These vulnerabilities are because by human errors as the complexity of implementation increases and new vulnerabilities are detected by researchers to check if SGX is vulnerable or not to unexplored attacks. Although these techniques are useful for hardening SGX.

Below are a few techniques [60, 61, 62, 63].

- **SGXElide**

  As we know SGX offers a secure enclave in that data and code are separated from the unprotected world, even from the underlying software and hardware. But by default, the enclave’s code can be disassembled before its initialization, and consequently, secrets cannot be embedded in the binary. As always developers wish to secure the enclave’s confidential data, SGXELIDE is a framework that permits code confidentiality via self-modification. Here the main thing is that code is considered as data and then dynamically gets back confidential data, post an enclave is created.

- **SGXBOUNDS**

  This technique provides shielded execution for legacy applications based on Intel SGX running on untrusted platforms to provide strong security guarantees. But memory attacks like heartbleed can give integrity and confidentiality properties and makes robust execution fully ineffective. So, to mitigate such kinds of attacks, memory safety methods can be applied for shielded execution. This technique helps to defend against software and hardware-based attacks. This technique helps in not exploiting the architectural properties of SGX. It uses a simple combination of compact memory layout and tagged pointers.

- **DR.SGX**

  This technique is used for hardening Intel SGX enclaves for preventing cache attacks with randomization of data location. As it has a weakness to software SCAs where adversaries can monitor CPU caches to get secret-based data access patterns. As far as known defense mechanisms have major limitations because it requires error-prone developers or suffers from extremely high runtime. And may only prevent specific kinds of attacks.

  This technique uses a randomized data locality as a prevention mechanism against SCAs which aims at data access patterns. The main aim of this technique is to interrupt the connection between the memory observation by a bad actor and the data access by the authentic user. The developers of this

DR.SGX have implemented it as a complier-based tool that permutes data location at fine granularity. For preventing the correlation of repeated memory access this technique periodically rerandomizes all the enclave data.

- **SGX-Shield**

  In general, execution environments deploy their ASLR to protect from memory corruption attacks. But SGX which is designed to serve security-sensitive applications on the cloud lacks this effectively. So, this technology enables ASLR for the SGX programs.

CHAPTER 5

5. Intel SGX for Client-Side and Server Side

5.1 Client Side

This is where my thesis started initially to understand the attacks on password managers on the client side. These are programs that are utilized to generate, store, and encrypt passwords for users. The only thing the user needs is to remember a master password and a username. Generally believed that using this password manager software will increase security but it is the main part for attackers nowadays. In 5.1.1 can see attacks on this software. Fault security to such software can lead to cost global economic losses.

Also, when large security breaches happen some thousand to millions of passwords may be exposed as they are stored in files, therefore people are suscept to password attacks like dictionary and rainbow table attacks [59].

5.1.1 Attacks on Password Managers

New potential attacks on password managers emerge day by day.

Some of them are below described:

- Adding salt to passwords is not at all enough to improve the security of the password from rainbow attacks as the reduced cost of the hardware made it feasible for the attackers to do brute force attacks [18] and a few tools like hashcat.

- Even storing hashed passwords will not stop attacks.

- Attacks on password managers show that strong security is not guaranteed in password managers which in turn leads to much other access.

- By exploiting the users' knowledge as they often use easy passwords which can be guessed.

- Such software seems too often lead to attacks to use phishing attacks and DDoS attacks.

5.1.2 Client-Side protection

In this section protection with Intel SGX with older versions of client devices. For this firstly, developers must allow client-side browser code to execute inside the enclave. Then the second point is that the client must share the CPU for integrity and confidentiality guarantees. So, client-side web language like JavaScript inside Intel® SGX could enable web applications to move trusted code to the client side. To be more trusted web pages let us consider the problem of input validation website should notify us whenever we enter an invalid input. Client-Side JavaScript will verify this and displays us the error message. Even though this all checks at the client side it is meaningless if we are not validated a second time on the server side.

But by using enclave validation is fully towards the client side, like inputs are sent to the enclave instead of the server. After the enclave validates it sends feedback to the client. Post all the values are correct the enclave can send them to the server using encryption values by the client. Using Trusted client-side web language enables functionality for the web applications otherwise it is impossible to achieve.

One important point about Intel SGX is its disappearance in the most recent generation for client processors which made an open discussion on the importance of client-side enclave. The fact Intel has deprecated SGX in its 11th and 12th generations of core processors [20] [21] for the client processors. SGX will only support new server platforms and new client devices (personal computers) SGX doesn't have support for.

5.1.3 Client – Side protection for deprecated devices

In this session protection mechanism for the client-side for SGX deprecated devices, i.e., latest devices. If the server-side has SGX support or not or any other TEEs at the server-side, these protection mechanisms might help new SGX client-side devices.

As client-side attacks result in the loss of customer data, business damage., etc., Mainly Personally identifiable information (PII) like credit card data and birth dates combined with usernames may be sold on the dark web for profits. This PII is crucial because it is used by the attacks for impersonation for their benefit [70]. In addition to this, there are OWASP client-side risks. These all attacks are happening because of failure to observe or stop web attacks.

Below are some of the best practices for client-side security:

- By regularly updating and patching all the applications and software that are associated with the websites.

- All websites must and should use a Secure Sockets Layer (SSL) certificate.

- When using third & fourth-party scripts should be very cautious while implementing them.

- Should use detection and identification technologies to scan for unknown threats, anomalies, and intrusions.

- By employing continuous monitoring and inspection tools along with a solution designed to get alert whenever any unauthorized script activity on the website.

- And more attention should be paid when it comes to patching, inspection, and monitoring.

- Sensitive data of the website should store appropriately may be in a unique meta field and should hide API keys from the public view.

There are many other protection mechanisms other than the above and even after all no one can help with zero-day vulnerabilities on the client side. So, more research and new technologies should emerge to protect the client side.

5.2 Intel SGX for Servers

Intel SGX is introduced for the Xeon server processor and is now available for servers including Infrastructure range and Intel Xeon E processor. By enabling this feature for servers, it gives a secure runtime environment by isolating part of the server’s physical memory known as a secure enclave. This way protects access to data that is being processed and the code that is being run. SGX protects data in use by using “hardware-based server security” and helps application developers protect selected code and data from modification; with enclaves, Intel SGX uses an isolated portion from the encrypted memory.

In 2017 December IBM announced access to offerings based on Intel SGX. Today SGX bare metal servers are present in all the regions on IBM Cloud. To provide SGX for servers, you need to select a bare metal server in the IBM cloud and can configure based on RAM, cores, frequency, etc. After these two steps, SGX needs to select “Intel Xeon E3-1270 v6”.

5.2.1 Data in Use Protection

An important usage of SGX is the encryption of data in use. Because in today's solutions for the data in transit, data in rest, and IAM helped enterprise data security. This is not end-to-end protection without a data-in-use solution [26]. This is because external attacks lead to internal incidents. After all, malicious internal incidents are on rising due to breaches in 2017. In this application, SGX data is ciphered and used without deciphering it. This provides a weak level of security as it exists only for the data in the use of encryption, and it is not practical for many applications like fully homomorphic encryption which provides little functionality like “order-preserving encryption”. Therefore, SGX looks like the best answer for this problem because all the data is ciphered under a key stored inside the SGX. After that to process data, it reads into SGX and internally decrypts, processes, and finally, the result is ciphered before it returns. This is how data cannot be decrypted outside the SGX and can’t be vulnerable.

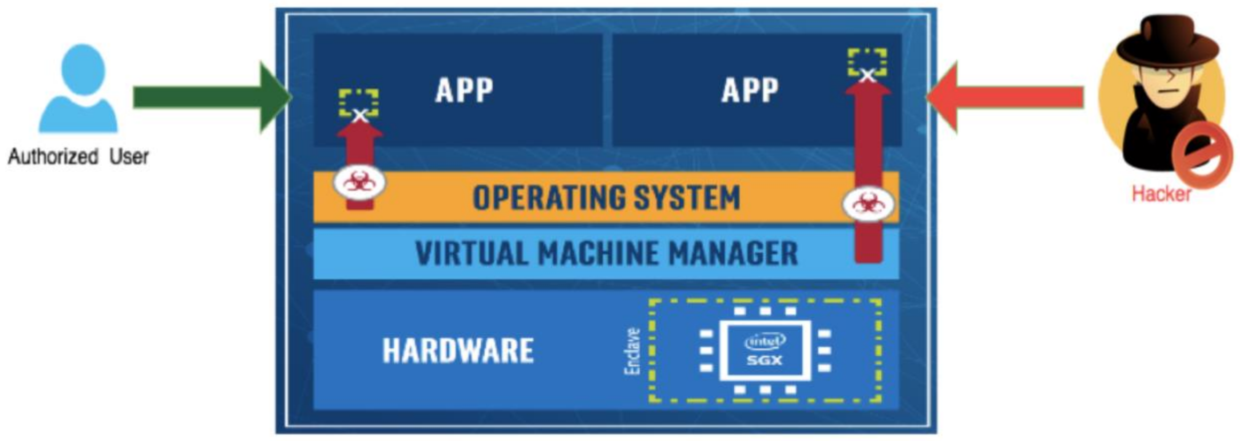

Figure 15 Data in use protection [78]

A notable use case is on cloud servers where we can encrypt things like SQL databases, and other data but cloud providers can never see the data in plaintext. This is achieved when the client uploads only an encrypted form of data to the cloud server and only entities of the SGX enclave should know the decryption key. Public key encryption cannot be used here as users who upload data must hold the same AES key. It is problematic but sounds perfect because AES-NI is only be applied securely, or else the decryption key could be derived by the bad actor and could steal the data by decrypting it.

One more main thing is that cache side channels are not just limited to extracting cryptographic keys but also their effectiveness against data processing. This is because cryptographic functions are restricted in “scope and implementations” must be made to make them strong against these cache attacks. In contrast to this, a normal application like a SQL operation is extremely data dependent. This is why a cache attack is used for the extraction of private data which is a disadvantage. Lastly, external memory access also appears to leak a lot of data in such applications. It can be solved by oblivious RAM.

Encryption is not always sufficient for ensuring privacy if the attacker can observe access patterns to the encrypted storage. They can learn important information like what our application is doing. By using oblivious RAM this issue can be resolved, here the memory is continuously shuffled as it is being accessed. Therefore, by fully hiding accessed data. This is good against a few attacks but affects the performance of SGX.

5.2.2 Privacy of Data

Data Privacy is protecting data that is in use, and encryption by considering many types of users with private data. This is one of the good applications of SGX. In this, all users who can carry out the computation can send encrypted data to the enclave, and then they get the output. Securing data privacy along with avoiding unwanted data disclosure is a big problem in the cloud when data analysis is done on mistrustful resources.

Present technology development in the trusted processor technology like Intel SGX has geographical efforts for performing in data analytics on a shared platform and where data security, the trustworthiness of these computations can be verified by the hardware. Even after these security features, a powerful adversary can infer private information from side-channel attacks like timing channels, cache access, and CPU usage by threatening user privacy and data [30]. However few studies and techniques proposed hiding sensitive information leaks by carefully designing data-independent access paths, but this technique will slow models with many parameters, mainly when used in present-day applications.

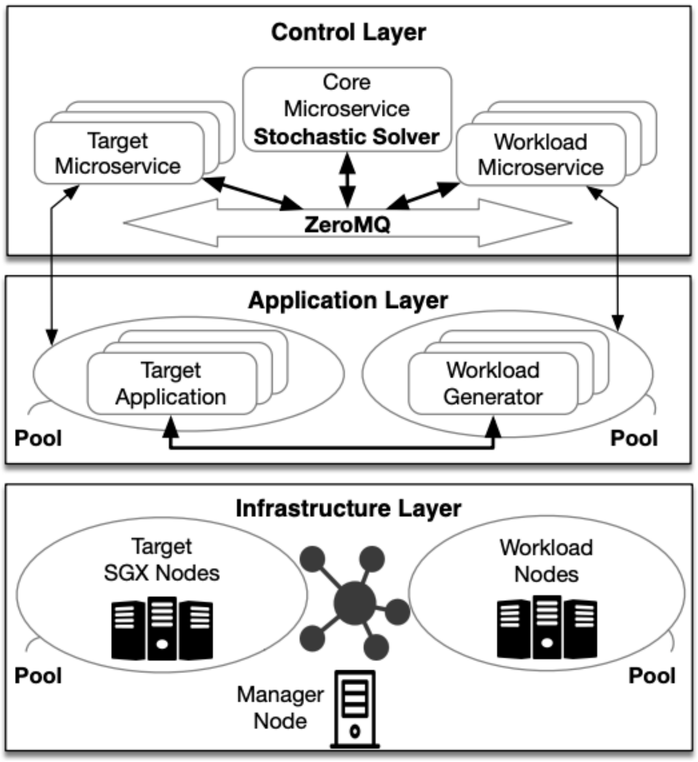

Figure 16 SGXTuner [27]

For preserving data privacy by using TEEs like Intel SGX, Mazzeo, and Giovanni [27], have designed a framework called " SGXTuner" along with end-to-end data encryption. They used a rule-based execution program for real IoT device data.

5.2.3 Data integrity in the cloud

We have many cloud storage services these days like AmazonS3, Dropbox, Google Drive, etc. And cloud service usage has become common these days so, verifying the integrity of the data on remote cloud turns out to be challenging. Because users have no control of their stored data on this cloud platform. So, to have users control their data to test the integrity confidentially and independently, there is a framework called EnclavePDP [65]. To test such features by using SGX SSL library and should port PBC libraries into SGX. And by using PDP schemes to make up the EnclavePDP framework. This enables users to verify the integrity of outsourced data with the cloud servers. Request to verify the integrity of the data encapsulated in TCP sockets and move to EnclavePDP which makes it easy to deploy EnclavePDP on the third-party cloud services

To verify integrity on behalf of the users using the existing PDP (Provable Data Possession) method which is mostly done by a TPA (Third Party Auditor) reduces the computation and communication burden of the users. But such things demand fully trusted TPA means the need to place TPA in TCB is not always reasonable. So, for this, we can use Intel SGX's EnclavePDP, which is an integrity verification framework for general and secure data.

CHAPTER 6

6. Conclusions and Future Work

6.1 Conclusions

This thesis gives a theoretical tutorial on Intel Software Guard Extension technology. Based on a few papers about vulnerabilities, applications and Intel documentation have covered the background of the technology and explained how to enable it and how can it provide security assurance with its security features. Firstly, it describes what is a secure enclave, it is an isolated and encrypted region of data and code. Then it describes the memory organization of the enclave. Intel designed this SGX to protect against software and hardware attacks which are called enclave-based security.

Secondly, it also describes other Trusted Execution Environments and what those technologies do. It is an environment for code execution where the execution of the code has a high level of trust as it protects against threats from the rest of the system. It is a technology that enables modern devices to provide a range of functionality by meeting software developers' requirements and this service provider cares about attestation, privacy, and other security features. TEEs also appear in IoT devices, smartwatches, smartphones, tablets, and even set-top boxes. Some use cases of TEEs are also mentioned like DRM and malware prevention and detection. These two applications can be implemented using SGX also.

Intel SGX has deprecated for its 11th and 12th generations this year for core client processors which means SGX is not supported in personal computers. It only supports new server platforms. The thesis analyzes SGX security and the general features of Intel SGX that provide security guarantees. They are hardware isolation, attestation (local and remote), encryption, and data sealing. As password managers are suspected of attacks as they store main sensitive data so described a few attacks on password managers can make use of Intel SGX for storing encrypted password files and for password processing to avoid attacks.

As SGX operates sensitive data it is prone to attacks so described few mitigation mechanisms. Mainly developers of the applications who make use of Intel SGX should be more careful because if the code inside the application has flaws all security guarantees are lost. To avoid side-channel attacks can use ORAM and some randomized algorithms. Whenever there is randomness, it is not that vulnerable to attacks and creates more protection.

Towards server-side Intel SGX as support so described two applications for server-side data privacy and data in use protection which uses encryption mechanisms.

Finally, other ways to get trusted execution on untrusted systems are described. Each has its features and issues, each with a different use case. Smart cards can be used either for cryptographic functions or as a general-purpose execution and they need to depend on another device for power. On another hand, TPM provides cryptographic operation on a coprocessor. The title of the thesis mentioned “An area of trust” and described possible attacks on Intel SGX by researchers, the title justifies this because there are no real-time cyber-attacks that happened.

6.2 Future Work

Although this thesis has reviewed SGX usage and limitations there are still some open research problems and challenges listed below:

- New research should try to prevent side-channel attacks on SGX by exploring a variety of defense mechanisms for different kinds of side-channel attacks.

- Secondly, exploring new undetected attacks on SGX could be a new research direction that will allow researchers and developers to build advanced protection techniques to magnify the security of Intel SGX.

- As SGX temporarily stopped supporting the new client-side devices more research on other TEEs to protect the client-side is necessary. As major attacks happen on the client side all the time.

- Even though there is no real-time attack recorded on the attestation feature practically it is possible to break that feature so a stronger mechanism can be researched and implemented in the future.

References:

[1] Kostiainen, Kari, Aritra Dhar, and Srdjan Capkun. "Dedicated Security Chips in the Age of Secure Enclaves." *IEEE Security & Privacy* 18.5 (2020): 38-46

[2] Ning, Zhenyu, et al. "Position paper: Challenges towards securing hardware-assisted execution environments." *Proceedings of the Hardware and Architectural Support for Security and Privacy*. 2017. 1-8.

[3] Costan, Victor, and Srinivas Devadas. "Intel SGX explained." *Cryptology ePrint Archive* (2016).

[4] Mofrad, Saeid, et al. "A comparison study of Intel SGX and AMD memory encryption technology." *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*. 2018.

[5] Valadares, Dalton Cézane Gomes, et al. "Systematic literature review on the use of trusted execution environments to protect cloud/fog-based Internet of Things applications." *IEEE Access* 9 (2021): 80953-80969.

[6] Chang, Rui, et al. "A trustenclave-based architecture for ensuring run-time security in embedded terminals." *Tsinghua Science and Technology* 22.5 (2017): 447-457

[7] Guan, Le, et al. "Building a trustworthy execution environment to defeat exploits from both cyber space and physical space for ARM." *IEEE Transactions on Dependable and Secure Computing* 16.3 (2018): 438-453.

[8] Feng, Xianglong, et al. "Towards the security of motion detection-based video surveillance on IoT devices." *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*. 2017.

[9] Kao, Chia Hung. "Survey on Evaluation of IoT Services Leveraging Virtualization Technology." *Proceedings of the 2020 5th International Conference on Cloud Computing and Internet of Things*. 2020.

[10] Lesjak, Christian, Daniel Hein, and Johannes Winter. "Hardware-security technologies for industrial IoT: TrustZone and security controller." *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2015.

[11] Silva, Leandro V., et al. "Security and privacy aware data aggregation on cloud computing." *Journal of Internet Services and Applications* 9.1 (2018): 1-13.

[12] Intel® Software Guard Extensions (Intel® SGX). "Intel".2017 "https://www.intel.it/content/www/it/it/architecture-and-technology/software-guard-extensions.html"

[**13]** Brasser, Ferdinand, et al. "Software grand exposure:{SGX} cache attacks are practical." *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. 2017.

[14] Burihabwa, Dorian, et al. "SGX-FS: hardening a file system in user-space with Intel SGX." *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2018.

[15] Weiser, Samuel, and Mario Werner. "Sgxio: Generic trusted i/o path for intel sgx." *Proceedings of the seventh ACM on conference on data and application security and privacy*. 2017.

[16] Karande, Vishal, et al. "Sgx-log: Securing system logs with sgx." *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. 2017.

[17] Götzfried, Johannes, et al. "Cache attacks on Intel SGX." *Proceedings of the 10th European Workshop on Systems Security*. 2017

[18] Brekalo, Helena, Raoul Strackx, and Frank Piessens. "Mitigating password database breaches with Intel SGX." *Proceedings of the 1st Workshop on System Software for Trusted Execution*. 2016

[19] Oleksenko, Oleksii, et al. "Varys: Protecting {SGX} Enclaves from Practical {Side-Channel} Attacks." *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 2018.

[20] Intel. "Https://cdrdv2.Intel.com/v1/Dl/GetContent/634648." *Intel*, Intel, May 2022, https://cdrdv2.intel.com/v1/dl/getContent/634648.

[21] Intel. “12 Generation Intel Core Processor.” *Intel.com*, Intel, Aug. 2022, https://cdrdv2.intel.com/v1/dl/getContent/655258.

[22] Nilsson, Alexander, Pegah Nikbakht Bideh, and Joakim Brorsson. "A survey of published attacks on Intel SGX." *arXiv preprint arXiv:2006.13598* (2020).

[23] Gjerdrum, Anders T., et al. "Performance principles for trusted computing with intel SGX." *International Conference on Cloud Computing and Services Science*. Springer, Cham, 2017.

[24] Dinh Ngoc, T., Bui, B., Bitchebe, S., Tchana, A., Schiavoni, V., Felber, P., & Hagimont, D. (2019). Everything you should know about Intel SGX performance on virtualized systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, *3*(1), 1-21.

[25] Brasser, Ferdinand, et al. "SANCTUARY: ARMing TrustZone with User-space Enclaves." *NDSS*. 2019.

[26] Lindell, Yehuda. "The security of intel SGX for key protection and data privacy applications." (2018): 16.

[27] Mazzeo, Giovanni, et al. "SGXTuner: Performance Enhancement of Intel SGX Applications via Stochastic Optimization." *IEEE Transactions on Dependable and Secure Computing* (2021).

[28] Islam, Md Shihabul, et al. "Secure IoT data analytics in cloud via Intel SGX." *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. IEEE, 2020.

[29] Zegzhda, Dmitry P., et al. "Use of Intel SGX to ensure the confidentiality of data of cloud users." Automatic Control and Computer Sciences 51.8 (2017): 848-854.

[30] Chandra, Swarup, et al. "Securing data analytics on sgx with randomization." *European Symposium on Research in Computer Security*. Springer, Cham, 2017.

[31] Randmets, Jaak. "An Overview of Vulnerabilities and Mitigations of Intel SGX Applications." *URL: https://cyber. ee/research/reports/D-2-116-An-Overview-of-Vulnerabilities-and-Mitigations-of-Inte l-SGX-Applications. pdf* (2021).

[32] Zheng, Wei, et al. "A survey of Intel SGX and its applications." *Frontiers of Computer Science* 15.3 (2021): 1-15.

[33] Kobayashi, Toshiki, et al. "Safes: Sand-boxed architecture for frequent environment self-measurement." *Proceedings of the 3rd Workshop on System Software for Trusted Execution*. 2018.

[34] Townley, Daniel, et al. "Composable Cachelets: Protecting Enclaves from Cache Side-Channel Attacks." *2022 USENIX Security Symposium*. 2022.

[35] Fei, Shufan, et al. "Security vulnerabilities of SGX and countermeasures: A survey." *ACM Computing Surveys (CSUR)* 54.6 (2021): 1-36

[36] Gai, Keke, and Meikang Qiu. "An optimal fully homomorphic encryption scheme." *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)*. IEEE, 2017.

[**37]** Zhang, Huanguo, et al. "Survey on cyberspace security." *Science China Information Sciences* 58.11 (2015): 1-43.

[38] Mahendran, Dinesh Raj, et al. "Trusted computing and security for computer folders." *International Journal of Medical Toxicology & Legal Medicine* 21.3 (2018): 83-86.

[39] Sabt, Mohamed, Mohammed Achemlal, and Abdelmadjid Bouabdallah. "Trusted execution environment: what it is, and what it is not." *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. IEEE, 2015.

[40] Bernard Ngabonziza, Daniel Martin, Anna Bailey, Haehyun Cho, and Sarah Martin. 2016. Trustzone explained: Architectural features and use cases. In 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC). IEEE, 445–451.

[41] Ruan, Xiaoyu. *Platform Embedded Security Technology Revealed*. Springer Nature, 2014.

[42] Kaplan, David, Jeremy Powell, and Tom Woller. "AMD memory encryption." White paper (2016).

[43] Brenner, Stefan, et al. "Securekeeper: Confidential zookeeper using intel sgx." *Proceedings of the 17th International Middleware Conference*. 2016.

[44] Chen, Yaxing, et al. "QShield: Protecting outsourced cloud data queries with multi-user access control based on SGX." *IEEE Transactions on Parallel and Distributed Systems* 32.2 (2020): 485-499.

[45] Condé, Rafael CR, Carlos A. Maziero, and Newton C. Will. "Using Intel SGX to protect authentication credentials in an untrusted operating system." *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018.

[46] Djoko, Judicael B., Jack Lange, and Adam J. Lee. "NeXUS: Practical and secure access control on untrusted storage platforms using client-side SGX." *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019.

[47] Fisch, Ben, et al. "Iron: functional encryption using Intel SGX." *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017.

[48] Frassetto, Tommaso, et al. "Jitguard: hardening just-in-time compilers with sgx." *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017.

[49] Fuhry, Benny, et al. "SeGShare: Secure group file sharing in the cloud using enclaves." *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020.

[50] Liu, Gao, et al. "SeDID: An SGX-enabled decentralized intrusion detection framework for network trust evaluation." *Information Fusion* 70 (2021): 100-114.

[51] Brasser, Ferdinand, et al. "Software grand exposure:{SGX} cache attacks are practical." *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. 2017.

[52] Schwarz, Michael, et al. "Malware guard extension: Using SGX to conceal cache attacks." *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, Cham, 2017.

[53] Van Bulck, Jo, et al. "Telling Your Secrets without Page Faults: Stealthy Page {Table-Based} Attacks on Enclaved Execution." *26th USENIX Security Symposium (USENIX Security 17)*. 2017.

[54] Wang, Wenhao, et al. "Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX." *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017.

[55] Hähnel, Marcus, Weidong Cui, and Marcus Peinado. "{High-Resolution} Side Channels for Untrusted Operating Systems." *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. 2017.

[56] Chen, Guoxing, et al. "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution." *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019.

[57] Kocher, Paul, et al. "Spectre attacks: Exploiting speculative execution." *Communications of the ACM* 63.7 (2020): 93-101.

[58] Van Bulck, Jo, et al. "Foreshadow: Extracting the Keys to the Intel {SGX} Kingdom with Transient {Out-of-Order} Execution." *27th USENIX Security Symposium (USENIX Security 18)*. 2018.

[59] Luevanos, Carlos, et al. "Analysis on the security and use of password managers." *2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*. IEEE, 2017.

[60] Bauman, Erick, et al. "Sgxelide: enabling enclave code secrecy via self-modification." *Proceedings of the 2018 International Symposium on Code Generation and Optimization*. 2018.

[61] Kuvaiskii, Dmitrii, et al. "SGXBOUNDS: Memory safety for shielded execution." *Proceedings of the Twelfth European Conference on Computer Systems*. 2017.

[62] Brasser, Ferdinand, et al. "DR. SGX: hardening SGX enclaves against cache attacks with data location randomization." *arXiv preprint arXiv:1709.09917* (2017).

[63] Seo, Jaebaek, et al. "SGX-Shield: Enabling Address Space Layout Randomization for SGX Programs." *NDSS*. 2017.

[64] Buchner, Nicolas, Holger Kinkelin, and Filip Rezabek. "Survey on Trusted Execution Environments." *Network* 21 (2022).

[65] He, Yun, et al. "{EnclavePDP}: A General Framework to Verify Data Integrity in Cloud Using Intel {SGX}." *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 2020

[66] Shepherd, Carlton, Raja Naeem Akram, and Konstantinos Markantonakis. "Remote credential management with mutual attestation for trusted execution environments." *IFIP International Conference on Information Security Theory and Practice*. Springer, Cham, 2018.

[67] Zhang, Xiaoyu. *Comparison of prominent trusted execution environments*. BS thesis. 2022.

[68] Guo, Yiran. "Research on the Video Copyright Protection." *2022 International Conference on Science and Technology Ethics and Human Future (STEHF 2022)*. Atlantis Press, 2022

[69] Shakevsky, Alon, Eyal Ronen, and Avishai Wool. "Trust Dies in Darkness: Shedding Light on Samsung's TrustZone Keymaster Design." *IACR Cryptol. ePrint Arch.* 2022 (2022): 208.

[70] Itkonen, Juuso. "How organizations can prepare for emerging threats from the dark web." (2022).

[71] Lipp, Moritz, et al. "PLATYPUS: Software-based power side-channel attacks on x86." *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021.

[72] Lantz, David. "Detection of side-channel attacks targeting Intel SGX." (2021).

[73] Qiu, Pengfei, et al. "VoltJockey: Breaking SGX by software-controlled voltage-induced hardware faults." *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST* IEEE, 2019.

[74] van Schaik, Stephan, et al. "SGAxe: How SGX fails in practice." *2020. https://sgaxe. com/files/SGAxe. pdf* (2020).

[75] Qiu, Pengfei, et al. "PMUSpill: The Counters in Performance Monitor Unit that Leak SGX-Protected Secrets." *arXiv preprint arXiv:2207.11689* (2022)

[76] “Intel® Software Guard Extensions Introductory Overview | Intel Software.” *Www.youtube.com*, www.youtube.com/watch?v=3MDIPAZnSTw.

[77] “Introduction to the Enclave Definition Language Intel® SGX | Intel Software.” *Www.youtube.com*, www.youtube.com/watch?v=IvQEplGe03k&t=47s.

[78] “Data-In-Use Protection on IBM Cloud Using Intel SGX.” *Archive of the IBM Cloud Blog*, 10 May 2018, www.ibm.com/blogs/cloud-archive/2018/05/data-use-protection-ibm-cloud-using-intel-sgx/.

[79] “Introduction to Trusted Execution Environment.” *Www.youtube.com*, www.youtube.com/watch?v=zshJnFms2xA&t=1074s.

[80] Lab, Systems Software and Security. “Enclave Layout.” *Systems Software and Security Lab*, gts3.org/pages/enclave-layout.html.

[81] “Secure Computation in Rust: Using Intel’s SGX Instructions with Teaclave and Fortanix.” *Www.notamonadtutorial.com*, 5 May 2022, www.notamonadtutorial.com/secure-computation-in-rust-using-intels-sgx-instructions-with-teaclave-and-fortanix/.

[82] Syed, Kamran, et al. “With the Help Of: 1. Intel SGX Tutorial (Reference Number: 332680-002) Presented at ISCA 2015 2.”

[83] Li, Wenhao, et al. “Research on ARM TrustZone.” *GetMobile: Mobile Computing and Communications*, vol. 22, no. 3, 17 Jan. 2019, pp. 17–22, 10.1145/3308755.3308761.

[84] Li, Wenhao, et al. “Research on ARM TrustZone.” *GetMobile: Mobile Computing and Communications*, vol. 22, no. 3, 17 Jan. 2019, pp. 17–22, 10.1145/3308755.3308761.

[85] “Confidential Computing and the Public Cloud.” *Brazilclouds.com*, www.brazilclouds.com/blog/confidential-computing-and-the-public-cloud.

[86] Mazzeo, Giovanni, et al. “SGXTuner: Performance Enhancement of Intel SGX Applications via Stochastic Optimization.” *IEEE Transactions on Dependable and Secure Computing*, 2021, pp. 1–1, 10.1109/tdsc.2021.3064391.