Miguel de Lemos Dias Rosa Anciães

Bachelor of Computer Science and Engineering

Trusted and Privacy-Enhanced In-Memory Data Stores

Dissertation plan submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Henrique João Lopes Domingos, Full Professor,
NOVA University of Lisbon

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

January, 2020

Abstract

The recent advent of hardware-based trusted execution environments provides isolated execution even from an untrusted operating system and possible hardware-based attacks. As the processor provides such "shielded" execution environments, their use will allow cloud users to run applications securely, for example on the remote cloud servers, whose operating systems and hardware are exposed to potentially malicious remote attackers and non-controlled system administrators' staff.

This dissertation will design, implement and evaluate experimentally a Trusted and Privacy-Enhanced In Memory Data Structure Store. The solution combines partial homomorphic encryption constructions, allowing that operations on the data store will be supported directly over in-memory encrypted data. Complementarily, the proposal will be designed to run on a trusted execution environment supported by Intel SGX technology, offering high availability for data store access, built-in replication, a LRU eviction model, support for transactions and options for on-disk persistence.

Keywords: Privacy-Enhanced Data Store; Homomorphic Encryption; Trusted Computing; Availability; Reliability.

Resumo

Independentemente da língua em que está escrita a dissertação, é necessário um resumo na língua do texto principal e um resumo noutra língua. Assume-se que as duas línguas em questão serão sempre o Português e o Inglês.

O *template* colocará automaticamente em primeiro lugar o resumo na língua do texto principal e depois o resumo na outra língua. Por exemplo, se a dissertação está escrita em Português, primeiro aparecerá o resumo em Português, depois em Inglês, seguido do texto principal em Português. Se a dissertação está escrita em Inglês, primeiro aparecerá o resumo em Inglês, depois em Português, seguido do texto principal em Inglês.

O resumo não deve exceder uma página e deve responder às seguintes questões:

- Qual é o problema?
- Porque é que ele é interessante?
- Qual é a solução?
- O que resulta (implicações) da solução?

E agora vamos fazer um teste com uma quebra de linha no hífen a ver se a LaTeX duplica o hífen na linha seguinte...

zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz comentar-lhe zzz zzzz zzz zzzz

Sim! Funciona! :)

Palavras-chave: Palavras-chave (em Português) . . .

Contents

1 Introduction 1

1.1 Context and Motivation . . . 1

1.2 Problem Statement . . . 2

1.3 Objectives and Planned Contributions . . . 3

1.4 Report Organization . . . 3

2 Related Work 5

2.1 Key-Value Stores . . . 6

2.1.1 Memcached . . . 7

2.1.2 Redis . . . 7

2.1.3 Amazon Dynamo DB . . . 8

2.1.4 Microsoft Azure Cosmos DB . . . 9

2.1.5 Microsoft Azure Cache for Redis . . . 9

2.1.6 Aerospike . . . 9

2.1.7 Discussion . . . 10

2.2 Trusted Computing Environments . . . 11

2.2.1 TPM – Trusted Platform Modules . . . 11

2.2.2 TPM - Enabled Software Attestation . . . 13

2.2.3 HSM – Hardware Security Modules . . . 13

2.2.4 Trusted Execution Environments . . . 14

2.2.5 Intel SGX . . . 14

2.2.6 Sanctum . . . 15

2.2.7 ARM Trust Zone . . . 16

2.2.8 Discussion . . . 17

2.3 TEE/SGX Enabled Key Value Stores . . . 18

2.3.1 Trusted Execution with Intel SGX . . . 18

2.3.2 Circumvention of SGX Limitations . . . 18

2.3.3 SGX-Enabled Secure Databases . . . 19

2.3.4 Discussion . . . 22

2.4 Related Work Balance and Critical Analysis . . . 23

3 3. Approach to Elaboration Phase 25

3.1 Refinement of Objectives and Contributions 25
3.2 System Model Approach 25
3.3 Planned Architecture and Implementation 25
3.4 Planned Testbench Environments 25
3.5 Relevant Evaluation Criteria 25

4 Workplan **27**

Bibliography **29**

Apêndices **35**

A Appendix 1 Lorem Ipsum **35**

B Appendix 2 Lorem Ipsum **37**

Annexes **39**

I Annex 1 Lorem Ipsum **39**

List of Figures

2.1 Azure Environment Integration 10
2.2 TPM insides 12
2.3 Remote Attestation Procedure 13
2.4 Arm TrustZone Stack [11] 17
2.5 Server-side components of EnclaveDB 20
2.6 Overview of ShieldStore 22

3.1 A figure with two sub-figures! 26

List of Tables

Listings

2.1 Redis Set & Get 6
2.2 How Fast is Redis 7

Glossary

aliquam
tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

computer
An electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals.

cras viverra
metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat.

donec nonummy
pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo.

integer sapien
est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus.

lorem ipsum
dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

maecenas lacinia
nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.

morbi ac
orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

morbi dolor
nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi.

nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo.

name arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo.

nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis.

sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Acronyms

ACL	Access Control List
AIK	Attestation Identity Key
AWS	Amazon Web Services
CA	Certification Authority
DBMS	Database Management System
DDoS	Distributed Denial of Service
ECALL	Enclave call
EK	Endorsement Key
EPC	Enclave Page Cache
GB	Gigabyte
HSM	Hardware Security Module
I/O	Input/Output
IoT	Internet of Things
KVS	Key-Value Store
LRU	Least Recently Used
LSM	Log-Structured Merge Tree
MB	Megabyte
MIM	Man-in-the-middle
OCALL	Out call

OS	Operating System
OTP	One-Time Password
P2P	Peer to Peer
PCR	Platform Configuration Register
SASL	Simple Authentication and Security Layer
SGX	Software Guard Extensions
SQL	Structured Query Language
SSL	Secure Sockets Layer
syscall	System call
TB	Terabyte
TCB	Trusted Computing Base
TCE	Trusted Computing Environments
TCG	Trusted Computing Group
TEE	Trusted execution environment
TLS	Transport Layer Security
TPM	Trusted Platform Module
USB	Universal Serial Bus
VM	Virtual Machine

SYMBOLS

*

Chapter 1

Introduction

In this chapter it's presented the context and motivation for this thesis, the main problem statement followed by the goals and objectives and all the planned contributions. In the end, it is presented the structure used in the following chapters of the document.

1.1 Context and Motivation

Cloud computing has gone through many steps that include grid and utility computing, application service provision and software as a service before reaching the level we know these days. The concept of delivering continuous resources through a global network is rooted in the 1960's. Some experts credit the professor and computer scientist John McCarthy [44] be proposing the concept of computation being delivered as a public utility.

Then, around 1970's the concept of the virtual machine (VM) started to gain popularity as it permitted multiple distinct computing environments to reside on one physical environment.

One of the first major cloud computing moments was the arrival of *salesforce.com* that pioneered the concept of delivering enterprise applications via a simple website. Later, around the 2000's, current big names like Oracle, SAP, Google, Amazon and Microsoft joined the trend and made the cloud world as it is today. [2] [3]

Over the past decade, cloud computing has evolve from something service providers told companies they should adopt to becoming the technology heart of not only major companies, but medium sized enterprises, small start-ups, personal projects and pretty much anyone who works in the computer science world.

Recent studies are foreseeing that 83% of enterprise workloads will be on the cloud by 2020 [1]. The array of services provided now are endless and the costs are attractive to businesses. These services allow developers to only pay for resource usage, and to take advantage of all the power of very large companies. Scalability at request, reliability with daily backups and seamless integration with a lot of other services are some advantages of moving to the cloud. And all of these functionalities without having to manage big infrastructures and a lot of servers, networks, disks, etc... [54].

All of this data and processing happening in someone else's machine started to raise privacy and security concerns. It has become a very attractive target for malicious hackers to attack cloud providers due to the amount of data they process and hold on their services. The best security researchers are always working with the providers to try and mitigate all bugs and vulnerabilities on their very large platforms which has become also a big attack vector. It has been reported by Microsoft, that *"There was a 300 percent increase in Microsoft cloud-based user accounts attacked year-over-year (Q1-2016 to Q1-2017)."* and *"The number of account sign-ins attempted from malicious IP addresses has increased by 44 percent year over year in Q1-2017."* [42]. Another example published on the Washington Post describes a sophisticated Man-in-the-Middle (MIM) cyber-attack that has targeted Apple's iCloud service in China, in an apparent attempt to collect user names, passwords and other private information. Also, Amazon Web Services has been in 2019 hit by a massive DDoS (Distributed Denial of Service) attack that kept the system down for about 8 hours straight, which can mean thousands of dollars lost by clients [15].

The well known Edward Snowden scandal [24], although not directly related, gave the world another perspective about the the security provided by the cloud providers, that keep the user's data secure from other hackers, but could technically be accessed by the respective provider, or by any system administrator with physical or remote access to the machines.

A previously mentioned study also reflects that *"66% of IT professionals say security is their greatest concern in adopting an enterprise cloud computing strategy"* [1].

1.2 Problem Statement

The problem behind the goals and objectives of this dissertation can be summarised in the following statements and questions:

Is it feasible to implement a solution for a remote cloud system with strong security features and policies with a good trade-off with performance? Can we combine and the performance of a key-value store with the security provided by an hardware based security solution? Can we isolate a system to the point where the hypervisor and operating system of the remote provider can be removed from the TCB? Is it possible to remove the threat of administrators of a cloud platform breaking data privacy? How do different types of replication affect performance and security?

1.3 Objectives and Planned Contributions

It is up to you, the student, to read the FCT and/or NOVA regulations on how to format and submit your MSc or PhD dissertation.

This template is endorsed by the FCT-NOVA and even linked from its web pages, but it is not an official template. This template exists to make your life easier, but in the end of the line you are accountable for both the looks and the contents of the document you submit as your dissertation.

1.4 Report Organization

It is up to you, the student, to read the FCT and/or NOVA regulations on how to format and submit your MSc or PhD dissertation.

This template is endorsed by the FCT-NOVA and even linked from its web pages, but it is not an official template. This template exists to make your life easier, but in the end of the line you are accountable for both the looks and the contents of the document you submit as your dissertation.

Chapter 2

Related Work

This chapter presents and briefly discusses the related work and the study performed beforehand in order to guide and give some context to the reader. It will present work that was used as the basis of this thesis, existent technologies and their relation with this project, and some comparisons between those existing technologies, the problem addressed in this thesis and the solutions proposed to solve, or better address, those very same problems.

First, in section 2.1 we explain and discuss for the first time the definition of a Key-Value Store. We present some use cases, current technology available, their differences and most importantly their security models and concerns.

Having discussed the software, section 2.2 will then address the environment on where that software will run on, most specifically the hardware. It explains and present the different ways to secure and authenticate the hardware, prevent hardware-based attacks and discuss some of the current products available and how they will be used across this thesis.

Section 2.3 will then make the bridge between software and hardware. It explains how Key-Value stores are currently being run on secure environments. It discusses how software and hardware work together to achieve a secure application. This chapter will be focused on the Intel SGX secure model and explain the advantages and disadvantages of this module.

To conclude the chapter, section 2.4 will combine the information of every sub-chapter and analyze it with a bigger perspective and better knowledge of the theme.

Along the next chapter we summarize the main relevant ideas that can be retained from each section for our objectives and expected goals.

2.1 Key-Value Stores

Key value stores are the simplest form of what computer scientists call a database. The simplicity lies on associating a value to a certain key and storing that pair, as well as retrieving the values of known keys. [31]

Listing 2.1: Redis Set & Get

```
redis> SET mykey "Hello"
"OK"
redis> GET mykey
"Hello"
redis>
```

Is this simplicity that makes this technology very attractive to developers. The ease of use, its high performance and speed are key aspects in favour of this technologies. However, simply working with keys and values might not be enough to more complex applications, and that is why Key-Value store product developers are introducing new features in order to make them appealing to a broader mass of users, always keeping them lightweight and fast.

For that lightweight and fast attributes, most of the key-value stores work in the computer memory. This allows fast get and write operations as opposed to persistent disk storage. Although, they work mainly in memory, most of the solutions offer some persistent mechanism so we can make use of its performance but still persist data in case of a disaster, server failure or any crash.

KVSs have been evolving for years and some are now more than a single key-value store module. A lot of them are now supporting a multi-model storage. Meaning that a value can be more than a single integer or a string. For example, Redis [45] as a multi-model store is not only a key-value store, but also [46]:

- **Document Store** - *"nonrelational database that is designed to store and query data as JSON-like documents"* [23]
- **Graph DBMS** - *"Graph databases are purpose-built to store and navigate relationships. Use nodes to store data entities, and edges to store relationships between entities"* [28]
- **Search Engine** - *"nonrelational database that is dedicated to the search of data content. Use indexes to categorize the similar characteristics among data"* [51]
- **Time Series DBMS** - *"Provides optimum support for working with time-dependent data. Each entry has a timestamp, the data arrives in time order and time represents a primary axis for the information."* [55]

So, the KVS world is becoming more and more versatile as the years pass.

In the next subsections its discussed and presented the overview of the current KVS technology. We picked the some top KVSs technologies nowadays according to db-engines [32] website.

2.1.1 Memcached

Memcached [37] is a free and open source key-value store released in 2003. It is described as a high performance distributed memory object caching system.

It is design to hold small chunks of data (strings and objects) to work as a cache for results of database calls, API calls, or page rendering. Its biggest use case is for use in speeding up dynamic web applications by alleviating database load.

This system lies on the simpler key-value store spectrum. It takes advantages of the simplicity of a key-value store to edge ease of development, and solving many problems facing large data caches. Its API is available for most popular languages. It has a LRU eviction technique which means that items will expire after a specified amount of time if not used.

When it comes to system replication, availability and reliability, Memcached has an interesting approach. In order to keep it blazing fast, there is no communication between server instances in a cluster. Memcached servers are unaware of each other. There is no crosstalk, no synchronization, no broadcasting, no replication. Adding servers will only increase the available memory.

As for its security context, Memcached spends very little, if any, effort in securing the systems for random internet connections. The servers only have support for SASL [49] authentication mechanism. This method of authentication is not implemented as end-to-end encryption, it only provides restriction access to the daemon, but it does not hide communications over the network. That means it is not meant to be exposed to the internet or to any untrusted users [38].

2.1.2 Redis

Redis [45] is an in-memory data structure store that can be used as a database, cache and also a message broker. Redis focuses on performance, so most of its decisions prioritize high performance and very low latency.

It has been benchmarked as the world's fastest database [47] and together with a their multi-model and its rich set of operations that can be performed over data it has been the leading key-value store according to use and popularity for a multiple set of years [32].

Listing 2.2: How Fast is Redis

```
$ redis-benchmark -t set -r 100000 -n 1000000
====== SET ======
1000000 requests completed in 8.78 seconds
50 parallel clients
3 bytes payload
keep alive: 1

99.59% <= 1 milliseconds
99.98% <= 2 milliseconds
100.00% <= 2 milliseconds
```

```
113934.14 requests per second
```

As said before, Redis is now not a simple KVS. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. It also has built-in replication, server side scripting, LRU eviction, concept of transactions and different levels of persistence. It provides high availability and automatic partitioning as well.

Redis provides replication in form of a master-slave model. This form of replication works with a single node (master node) where all writes occurring will be replicated to the other Redis instances (slave nodes). Writes on nodes other than the master will not be replicated. Redis provides a read-only setting that can be applied to slave nodes to prevent states differences between instances.

Security is not Redis' primarily concern (just like others). *"In general, Redis is not optimized for maximum security but for maximum performance and simplicity"* [48]. It is design to be access by trusted clients inside trusted networks. This means that it is not supposed to be publicly exposed. Redis implements a simple authentication system with a password on the configuration file for client authentication. It is also advised to run it behind a proxy to enable some ACL policies and SSL network security.

There are a few other security concerns that Redis addresses, but has we can now start to see, in this types of stores, security falls behind performance and usability.

2.1.3 Amazon Dynamo DB

Amazon Dynamo DB [6] is a fully managed NoSQL database service. It is a key-value store and a document store that is built based on the dynamo paper [22]. This paper describes a P2P (peer-to-peer) network with high availability, eventual consistency and very easily scalable. It also successful handles server and data center failures and network partitions.

Amazon builds on this paper and offers DynamoDB as a service in their platform. It is a hosted system in the Amazon Web Services [8] infrastructure and it is fully managed. That means no need for low level server configurations or maintenance. It is all managed by the AWS team and offered to the user with a nice configuration interface. It also means that it has built-in security, backup and restore and in-memory caching for internet-scale applications. It also offers seamless scalability by increasing the number of nodes/servers according to current traffic received by the application on any given time.

This technology focuses more on high availability but also achieves very high performances and very low latency and being fully managed it also takes advantages of the AWS infrastructure full power. It currently sits second on the db-engines [32] most popular ranking.

2.1.4 Microsoft Azure Cosmos DB

Microsoft Azure Cosmos DB [40] is a fully managed database service provided by Microsoft Azure [41]. This service provides a global distributed, horizontally scalable, multi-model database. Its multi-model architecture can work as a key-value store, a Document Store, a graph DBMS and a wide column store.

It's very proud and excels in the easy of global scale with the system call *Turnkey global distribution*, providing transparent multi-master replication and a set of users configurable consistency options. It also strongly advertises a *Multi-Model Multi-Api* feature where you can use multiple data types on this single database service. Cosmos DB automatically indexes all data and allows the user to use various NoSQL APIs to query the data.

As a fully managed service, Cosmos DB makes use, in the background, of the large infrastructure with almost unlimited resources and capabilities provided by Microsoft, which means it also has built-in security, fail-over mechanisms for disaster recovery, and high performance with single digit read and write latencies.

2.1.5 Microsoft Azure Cache for Redis

Microsoft Azure Cache for Redis [39] is a service provided by Microsoft Azure that joins the open source world of Redis with the commercial side of a fully managed and hosted platform.

It uses at its core the Redis server technology and provides ease of deployment and management, built-in global replication, Azures' infrastructure security and flexible scaling and Redis superior throughput and low latency performance.

Being in the Azure ecosystem provides nice integration with all Azures' services as shown in figure 2.1.

2.1.6 Aerospike

Aerospike [4] is an enterprise-grade, high performance Key-Value Store. It is another KVS technology currently available today. It promises a philosophy of *"no data loss"* through Strong Consistency. Normal systems trade requiring this type of consistency usually trade performance for data integrity but Aerospike allows it with minimal performance loss. That means it can be used for example in banking payments, retail and telecommunications use cases.

It also provides a dynamic cluster management and unique flexible storage. That enables very easy deployments and particularly very easy scalability, so it is able to meet any data volume needs and still maintaining low latencies across that wide range of data volumes, from low volumes until hundreds TB of data.

As for security, it includes (the enterprise version) a database access management and audit trail logs. It also includes transport level encryption for client-server traffic and

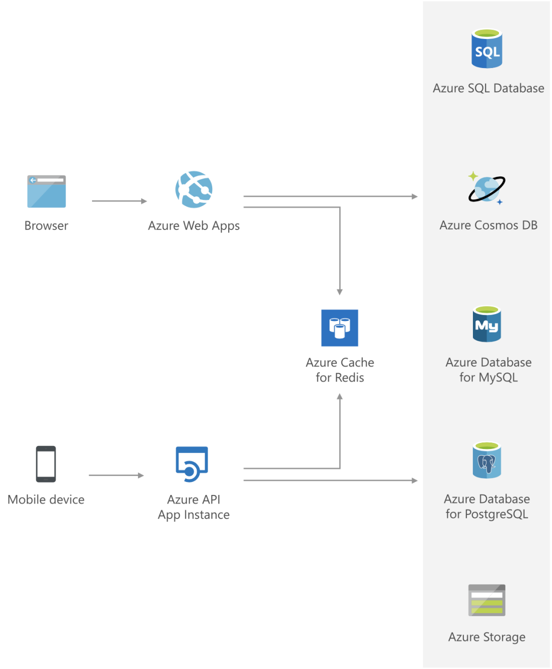


Figure 2.1: Azure Environment Integration

cross-datacenter traffic [5].

2.1.7 Discussion

In this chapter when gather information about the overview of th current Key-Value Store. We can conclude that the most important feature of this technology is the performance and all of the above products mentioned do focus on that characteristic. Some of them even compromise in another features to achieve the best performance possible. Security is not the main concern and the most used measures in the current technologies being securities implementations at the network and transport level by using TLS and also full disk encryption.

Network and transport layer security is a must when implementing any system, and this thesis will also use those standards.

As for full disk encryption on the server, it opens up some attack vectors. Full disk encryption means that random users will not be able to query the data but credentialed users can. Although, anyone with full access to the database, for example database operators or/and administrators, can decrypt and access all information. This creates a risk of privacy breaking due to hackers wielding stolen credentials, rogue insiders who have been granted more access than they need or the well known honest-but-curious adversary model, where an administrator with full credentials does not have bad intentions, but, driven by curiosity, access information therefore breaking data privacy. A cloud based KVS service like the ones talked above, this type of vulnerabilities can be a major concern for a use case with very sensitive data since the server would be off premises, there is no control over it when it comes to privacy of data.

This thesis will implemented a system based on Redis, the most popular and used Key-Value Store currently used and will try to solve some of the problems with security described above. It will compare the principle feature of a KVS, the performance, of a simple and normal Redis server and a privacy-enhanced Redis solution so the user can calculate the trade-off between performance and security and applied the correspondent solution to their own use case.

2.2 Trusted Computing Environments

Modern data processing services hosted in the cloud are under constant attack from malicious system administrators, server administrators and hackers who exploit bugs on applications, operating systems or even the hypervisor. However, current days shows a massive trend of business moving to the cloud infrastructure looking for easy deployment, managed services with built-in replication and fault tolerance, fast and trivial scaling and predicted costs.

With more and more data exposed in the cloud, hackers have a bigger desire to exploit and look for vulnerabilities. This results in frequent data breaches that reduce trust in online services. The need for cloud providers to ensure a level of security and trust to make the user comfortable of moving its data to the cloud has never been bigger, and with that need some solutions in the form of Trusted Computing Environments (TCE) appeared.

Trusted Computing is a concept that strives to provide strong confidentiality and integrity guarantees for applications running on untrusted platforms. It forces a certain machine to behave an expected way even if running on a remote or machine that is out of our control.

TCE will also provide a decrease of the Trusted Computing Base (TCB) - the amount of components that the application needs to trust in order to run smoothly. By isolating the service running on this trusted environments (limiting the set of instructions available and encrypting data), it prevents the operating system, the hypervisor and even malicious system administrators (three components normally on the TCB) to break data confidentiality and integrity within this environments.

There are a few hardware/software based solutions to achieved a trusted computing environment, and they will be explained in the next sections.

2.2.1 TPM – Trusted Platform Modules

A Trusted Platform Module, also known as a TPM is a technology proposed by the Trusted Computing Group (TCG) designed to provide hardware-based security related functions. It's a chip embedded into the motherboard and includes multiple security mechanisms to make it tamper resistant to physical harm and malicious software is unable to mess with its security features [56]. Some key advantages of using TPMs are:

- Generate, store, and limit the use of cryptographic keys
- Platform identity by using the TPM's unique RSA key, which is burned into itself also known as Endorsement Key (EK) and never leaves the TPM.
- Help ensure platform integrity by taking and storing security measurements.

Figure 2.2 shows the mains components and services provided by a TPM module. As shown in the figure, all of them only have one access point I/O which is protected and safely managed by the TPM execution engine.

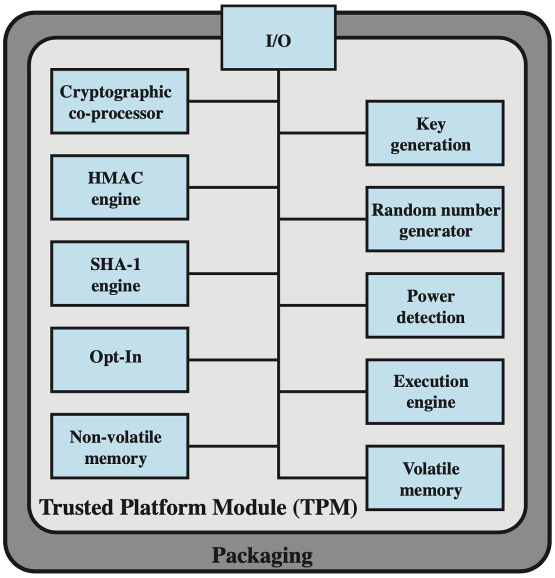


Figure 2.2: TPM insides

With all the components described by figure 2.2, TPMs provide there main TPM features: Encryption, Authenticated Boot and Attestation.

The first feature is used for every security and confidentiality aspects, mainly generating cryptographic keys, encrypting, signing and hashing data with secure standard algorithms melted in the module.

Authenticated Boot is the ability to boot the OS in stages, assuring that each portion of OS, as it is loaded, is a version trusted and approved for use, detecting hardware and software changes on every stage to verify if the code loaded can be trusted. This boot sequence happens with the help Platform Configuration Registers (PCR) that store the trusted software hashes.

The attestation feature is a way for a client to remotely check the state of a machine and will be further explained in the next subsection.

2.2.2 TPM - Enabled Software Attestation

The remote attestation feature of a TPM is the ability of a program to authenticate itself against external verifiers. Is a mechanism that allows a remote party to verify the internal state of the OS or another software and decided whether or not that piece of software is intact and trustworthy. The verifier can trust that the attestation data is accurate and not tampered with because it is signed by the internal key of the TPM, a special key known as the Attestation Identity Key, known from now on as AIK [14].

A remote attestation procedure is described in image 2.3 [13]:

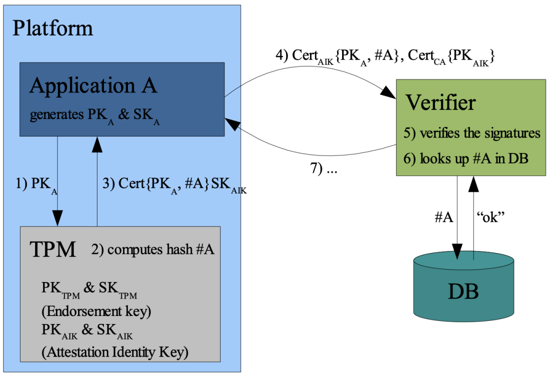

Platform
Application A
generates PK_A & SK_A
1) PK_A
3) Cert{PK_A, #A}SK_{AIK}
TPM
2) computes hash #A
PK_{TPM} & SK_{TPM}
(Endorsement key)
PK_{AIK} & SK_{AIK}
(Attestation Identity Key)
4) $Cert_{AIK}${PK_A, #A}, $Cert_{CA}${PK_{AIK}}
Verifier
5) verifies the signatures
6) looks up #A in DB
7) ...
#A
"ok"
DB

Figure 2.3: Remote Attestation Procedure

1. The application "A" generates a public/private key pair PK_A & SK_A and asks the TPM to certify it.

2. The TPM computes a hash value #A of the executable code of program "A".

3. The TPM creates a certification including PK_A and #A and signs it with the attestation identity key SK_{AIK}.

4. When application "A" wishes to authenticate itself to a remote party, it sends the cert. of its public key and hash value #A along with a cert. issued to the TPM by a trusted certification authority (CA).

5. The remote party verifies the cert. chain.

6. The remote party looks #A up in a database which maps hash values to trust levels.

7. If application "A" is deemed trustworthy, we continue the communication, probably by using PK_A to establish a session key.

2.2.3 HSM – Hardware Security Modules

An (HSM) hardware security module is normally an external module that can be added to a system, in form of a USB device or a component living in a secure network as a trusted server, instead of being embedded into the motherboard like a TPM. It provides a dedicated system of hardware enable accelerated cryptographic functions like encryption, decryption, key generation and signing capabilities [29]

What makes this devices so secure, like the TPM, is it can´t be interfered with by external code, and it provides an array of protective mechanisms to detect and prevent external physical tampering like drill protection foil, resin-embedded chips as well as temperature and voltage sensors. Any detection of tampering will result in an alarm as well as countermeasures by the applications installed inside. [19].

HSM can have various applications and can be used in simple forms for example a specific bank dongle that generates OTP (one-time password) for accessing your account or be a big corporation and enterprise appliance in various industries, e-health, automotive and IoT systems.

2.2.4 Trusted Execution Environments

A Trusted Execution Environment (TEE) is an abstraction that describes a machine capable of executing a given program P in isolation, i.e. whose output is determined by the initial state of P and a set of defined inputs given into the TEE (Barbosa et al., 2016).

It is a secure area of the main processor that ensures sensitive data and code loaded inside is stored, processed and protected in an isolated and trusted environment. As such, it offers protection from software attacks even the ones generated in the operating system.

A TEE guarantees that:

- The code loaded in the environment is authentic and was not tampered by an attacker.

- All system state is correct (CPU registers, memory and sensitive I/O).

- The code, all data generated and runtime state is confidential and stored persistently.

The threat model of a TEE should include all software attacks and the physical attacks performed on the main memory and its non-volatile memory.

"There are many interpretations of what is meant by Trust. In the TEE it is used to imply that you may have a higher level of trust in validity, isolation and access control in items (assets) stored in this space, when compared to more general purpose software environments"[58].

2.2.5 Intel SGX

"Intel® Software Guard Extensions (Intel® SGX) is a set of instructions that increases the security of application code and data, giving them more protection from disclosure or modification."[30].

These set of instructions are one of the latest iterations of trusted computing solution and designs and tries to tackle the problem of securing remote computations by leveraging secure hardware on the remote host machine. The SGX processor enables a secure container called enclave which protects the confidentiality and integrity of the execution, such as code and data while relying on software attestation mechanisms.

A SGX can be though as a reverse sandbox. With a sandbox you are trying to protect the system from your application, but with SGX you are trying to do the opposite and protect the application from the system. The system can be the OS, the hypervisor, the BIOS, the firmware or the drivers [53].

It provides this kind of security from the hardware by isolating all the private data from the outside, placing them into the EPC (Enclave Page Cache), which is memory zone with guaranteed access management by the CPU where it will deny every external access and only allow access through the associated enclave. This region of the memory is also known as the private/protected memory or the trusted part of the application.

A SGX enabled application is broken into two parts, the untrusted and trusted parts. The trusted part of the application is all the processing that deals with any sensitive data the application is handling. This part will be run inside enclaves and be stored in protected memory The rest will live in normal memory and not be protected.

The SGX will create an enclave when sensitive code needs to run by a specific SGX CPU instruction and will load the sensitive code into an EPC inside the protected memory. Once all pages are loaded into the EPC, and the loading in complete, an authentication hash is computed and is available for remote attestation so an user can verify that the code running in the enclave has not been tampered with.

When running, the execution always happens in protected mode, and to prevent data leaking, the CPU will not directly address an interrupt, fault or VM exit, but will instead emit another specific instruction to properly exit the enclave, save CPU state into the enclave and only then will service the fault.

With all this properties, Intel® through SGX set of instructions and implementation tries to achieve a secure and trusted environment with guarantees of code and data isolation, confidentiality and integrity from attackers such as the OS, hypervisor, any hardware and even physical attacks [20].

2.2.6 Sanctum

Sanctum [21] is an open-source project that shares the same as goal as Intel SGX, providing strong provable software isolation to protected the data from external hardware and software, but claims to be simpler and protect against indirect attacks called side channel attacks [33] such as cache timing attacks [18] that have been know to exist in SGX [27] [50]. These are additional software attacks that can infer private information by analyzing a program's memory access patterns.

Following the minimal and simple concepts, it uses minimal invasive hardware and it does not required any modifications to the CPU major blocks, but only adds hardware to the interfaces between blocks. This allows for a respectable overhead by maintaining normal clock speeds as it does not modify the CPU core critical execution path.

Sanctum project builds on the SGX programming model and implements an architecture that deviates as little as possible from the one built by Intel. Although it differs from SGX by implementing the enclaves via a small combination of hardware extensions to RISC-V (an open source set of CPU instructions [57]) and a trusted piece of software called the security model, as SGX implements them via hardware microcode and presents a set of CPU instructions to manage the enclaves.

This security monitor is the core of the project and configures the hardware to enforce low-level rules that controls the enclaves's access policies. As explained in the Sanctum paper, "*the security monitor checks the system software's allocation decisions for correctness and commits them into the hardware's configuration registers*". One of the examples and the main points of upgrade compared to SGX is that Sanctum keeps the enclave page tables inside enclaves memory, protecting the system against the timing attacks referred above by keeping the page table dirty and accessed bits private. Their hardware extensions make sure that enclaves page tables only point to enclave memory and untrusted OS tables only point to Os memory regions and never to enclave private memory.

Sanctum is also open to the public which makes easier for security researchers to audit and find vulnerabilities and to further encourage the analysis of the code, Sanctum security monitor is written in portable C++ code and can be used across different CPU implementations.

2.2.7 ARM Trust Zone

ARM Trust Zone [10] is a technology that offers a system wide approach to security based on hardware enforced isolation built into the CPU [9]. The principle of the technology is to separate the trusted and untrusted by two virtual processors backed by hardware access control. The two states are referred as worlds, where the first is called the secure world (SW) and the other is the normal world (NW) like figure 2.4 shows.

The non-secure world (or normal world) is where the OS and most of the software and applications will be running, as for the secure world is where more secure and sensitive software will run and will ensures that vital information is not intercepted by a third party. The security is enforced because each of the worlds acts isolated from the other as a runtime environment with separated resources such as memory, processor, cache, controllers, interrupts. The ARM hardware has separate copies of the registers for each worlds and cross-world register access is blocked. However the Secure Monitor shown in figure 2.4 can access non secure registers while running in secure world. This means that the monitor can then implement context switching between both worlds.

When in Normal World, the application calls a specific ARM instruction call SMC

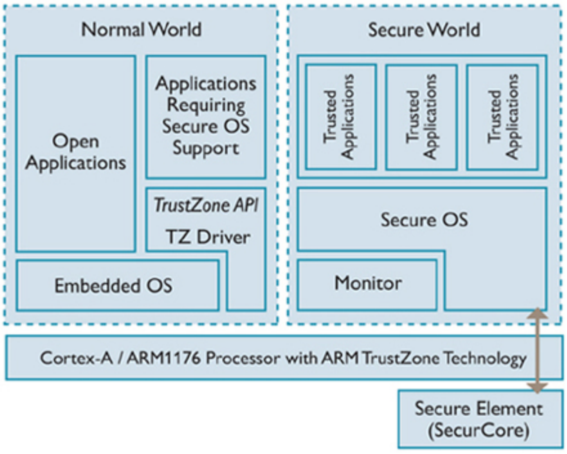


Figure 2.4: Arm TrustZone Stack [11]

(Secure Monitor Call) to call back inside the secure world and execute in code in a secure manner.

By keeping the worlds separated from each other, the ARM TrustZone can keep applications running in secure mode isolated from the normal world applications such as the OS and thus achieving another implementation of a TEE.

2.2.8 Discussion

The TCB (Trusted Computing Base) is the of set computer components (hardware), software and data that we need to trust and deem as not malicious in order to use a system. It's a group of various elements that are critical to a systems security in a way that any bug or vulnerability occurring from inside the TCB components might compromise the security and privacy of the entire system. On the other hand, security flaws and bugs from outside of the TCB should not become a security issue.

Hardware security like TPMs and TEE technologies have the goal to drastically reduce the TCB of a system, for example, remove the OS from the trusted base to make sure that a remote compromised machine vulnerability does not affect the security and privacy of a application or system.

By isolating the program from outside and uncontrollable sources like cloud infrastructures OSs, hypervisors and hardware we can more safely deploy sensitive applications to those cloud providers. That is the goal of all different implementations of trusted execution environments like the Intel SGX and the ARM TrustZone.

Although not free of some problems, SGX implementation of a TEE seems to be the most accepted technology, with serious and skilled developers and security researches always working to mitigate any vulnerability in order to create a truly trusted and isolated environment.

2.3 TEE/SGX Enabled Key Value Stores

There has been an increase trend from developers to move their applications to the cloud. It provides dynamically and almost seamlessly scaling with predict cost. Although it also means that users need to rely on the cloud providers for securing and maintaining the integrity of their applications. That means the user must trust not only the provider's staff but also its globally distributed software and hardware not to expose their private data. Today's cloud providers only aim to protect their privileged code from the untrusted code (the user's code) and do not provide any guarantees about the opposite scenario.

To mitigate this use case, and after studying and discussing the Key-Value stores technologies and also the trusted platform modules as well as the trusted execution environments, in this chapter, it will be presented how are this two topics being combined and used together.

It will be more focused on the Intel SGX platform as it is the one that will be used throughout this thesis. Currently, there are a number of databases who leverage this technology to provide a more secure environment and service. In the this chapter it's presented how they work and operate, discussed the differences between them and also the how the work planned to be performed on this thesis will solve some of the problems and caveats.

2.3.1 Trusted Execution with Intel SGX

As explained before, Intel SGX provides a trusted execution environment by running code inside the enclaves. It creates an isolated environment where we can run some instructions as securely as possible, without OS intervention.

Key-Value Stores and other database type systems can leverage this secure and isolated environment to perform queries on very sensitive data that would otherwise be vulnerable to some attacks. There are a few techniques currently implemented to use isolated environments. Maintaining an encrypted database and using enclaves cryptographic capabilities to decrypt data and perform queries on plain text with the assurance of no data leaking is a possible use case. Also, maintaining a database fully on enclave memory, where it cannot be access by anyone other than the CPU is another way to keep the data secure by leveraging isolated and trusted execution environments. Different techniques will be furthermore discussed bellow.

As we can see, isolated and trusted execution environments are an important feature when it comes to protecting the data from the OS and Key-Values Store systems do benefit from them.

2.3.2 Circumvention of SGX Limitations

There are a few limitations and challenges of the SGX platform that we address when programming for such technology.

It starts with a big challenge of choosing and defining what parts of the program can benefit of the SGX security. As it is known, it works with two major application components, the trusted and untrusted modules of or program. The limitations have to be thoroughly analyzed so we can make that definition.

The main limitations are:

- Performance
- Memory
- I/O
- syscalls

In the KVS world, as we extensively covered, performance is the major concern and there is no real way around this limitation. Using secure enclaves will definitely decrease the supposed blazing fast performance. Although, with intelligent partition between the untrusted code, which will be fast, and the trusted instructions, which will be slower we can limit the performance overhead. By separating and well defining both modules of the application, we can decrease the code that needs to run securely and find a fine compromised between security and performance.

Memory sizing is also a limitation when using enclaves in SGX technology. The amount of private secure data that can be maintained by the enclave is limited to the size of the enclave cache, which is around 128 MB, being that only about 94 MB are available to the application, with the rest reserved to metadata. Now, with SGX v2 and for some operation systems, mainly Linux because of paging swap support, it can be increased up to all the memory available in the system [52] by swapping pages from the EPC to main untrusted memory, with guaranteed of confidentiality, integrity and data freshness. Although, page eviction to main untrusted memory introduces a big overhead because of encryption and decryption and integrity checks (2x - 2000x) [12]. Clever partitioning of the application into the untrusted and trusted modules will help to overcome this limitation as described in the next sections.

I/O and syscalls are limited by default on the enclave for security purposes, so it can't affect or be affected by the OS. There is a way to perform and access I/O and syscalls through the aforementioned ECALLs and OCALLs (section 2.2.5 of this thesis), but they have to be accounted for when implementing the application.

2.3.3 SGX-Enabled Secure Databases

Database management service developers are now implementing secure databases ready to take advantage of Intel SGX hardware. It differs from normal databases because it runs on top of protected and encrypted memory so it can work with minimal TCB.

Next subsections present and discuss the overview of the current technology that leverages SGX to provide a secure database.

2.3.3.1 EnclaveDB

EnclaveDB [43] is a privacy enhanced and secure database that works alongside with Intel SGX and provides an Structured Query Language (SQL). It uses its technology to maintain all sensitive information inside SGX enclaves in order to keep them secure from an threat model of strong adversaries that can control the entire software stack on the database server. It resists attack from the administrator server, the database administrator and attacker who may compromised the operating system, the hypervisor or the database server.

Following Intel's application guidelines, EnclaveDB has an two part architecture: trusted (running on the enclave) and untrusted modules. The enclave hosts a query processing engine, natively compiled stored procedures and a trusted kernel which provides API's for sealing and remote attestation. The untrusted host process runs all other compo- nents of the database server. Figure 2.5 shows the architecture of the enclaveDB server-side.

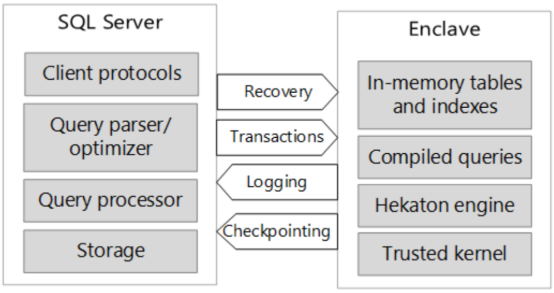


Figure 2.5: Server-side components of EnclaveDB

Leveraging TEE, EnclaveDB then provides a database with a SQL interface and guarantees confidentiality and integrity with low overhead. With its design it also reduces the TCB to a smaller set than any other "normal"database.

2.3.3.2 Pesos DB

Pesos [36] is a secure implementation of object storage services like Amazon S3 [7], Azure Blob Storage [16], Google Cloud Storage [26] among others. In these current large-scale services, due to their complexity, the risk of confidentiality and integrity violations increase significantly. This storage systems are characterized by multiple layers of software and hardware stacked together which means the access policies for ensuring confidentiality and integrity are scattered across different code paths and configurations, thus exposing the data to more security vulnerabilities. Furthermore, untrusted third-party cloud platforms expose an additional risk of unauthorized data access by a malicious administrator.

Pesos allows clients to specify per-object security policies concisely and separately from the remaining storage stack. It also provides cryptographic attestation for the stored objects and their associated policies to verify the policy enforcement.

It enforces this policies by leveraging the Intel SGX for trusted execution environments and Kinetic Object Storage [35] for trusted storage (secure storage - not the focus of this thesis). It structures a policy-compiler, its binary-format interpreter, per-object policy metadata, and the enforcement logic into a single layer of the storage stack. With this unification, it drastically reduces the TCB when compared to the order cloud services. Then it uses the trusted execution environment provided by SGX to connect directly Kinetic disk through an encrypted Ethernet connection allowing for object transfer and policy enforcement securely without any intermediate layers in the storage stack.

2.3.3.3 Speicher

Speicher [17] is a secure LSM-based Key-Value store that uses Intel SGX and it ensures not only strong confidentiality and integrity properties, but also data freshness to protect against rollback/forking attacks. It leverages SGX technology to achieve those security characteristics focusing on providing a **persistent** service, tolerant to system faults and securely recovering from crashes. It also tackles in interesting ways, two of the major limitations of SGX: Memory Limits and Performance.

Implementing a Key-Value Store has a major requirement - High performance and low latency queries for big data structures. As already discussed, SGX has some memory limits. The enclave memory is located in the Enclave Page Cache (EPC) which is limited to 128 MB with about 94 MB available for application use (the rest being reserved for metadata). To allow creation of enclaves with bigger size than EPC, the OS can use secure paging mechanism where it evicts pages to untrusted memory. Although with page encryption, decryption and integrity checks, this solution introduces high overheads (2× - 2000×) [12].

To address this performance and memory problems, the developers of Speicher implemented the following custom features (from Speicher public paper):

- *"**I/O library for shielded execution**: Direct I/O library for shielded execution. The I/O library performs the I/O operations without exiting the secure enclave; thus it avoids expensive system calls on the data path."*

- *"**Asynchronous trusted monotonic counter**: Trusted counters to ensure data freshness. The counters leverage the lag in the sync operations in modern KVS to asynchronously update the counters. Thus, they overcome the limitations of the native SGX counters."*

- *"**Secure LSM data structure**: Secure LSM data structure that resides outside of the enclave memory while ensuring the integrity, confidentiality and freshness of the data. Thus, the LSM data structure overcomes the memory and I/O limitations of Intel SGX."*

The technology leverages SGX with a clever partition between trusted and untrusted modules of the application. By maintaining the encrypted data on untrusted memory hardware it addresses the memory and persistent limitations, and by keeping some information in secure enclave memory and with a good I/O library it overcomes (to an extent) the performance issues.

2.3.3.4 ShieldStore

ShieldStore [34] is a *"(...) shielded in-memory Key-Value Storage with SGX"*. It aims to provide a very fast and low latency queries over very large data trying to overcome the SGX memory limitation. It accomplice's it by maintaining the majority of the data structures in the non-enclave memory region, addressing as well the performance issue by not relaying on the page-oriented enclave memory extension provided by SGX.

ShieldStore runs server-side in the enclave to protect encryption keys and for remote attestation and it is used to perform all the KVS logic. It uses a hashed index structured but places it in the unprotected memory region instead of the enclave EPC. As the main data structure is not protected by the SGX hardware, each data entry must be encrypted by ShieldStore in the enclave, and written to the main hash table.

The main flow and architecture is as described on figure 2.6.

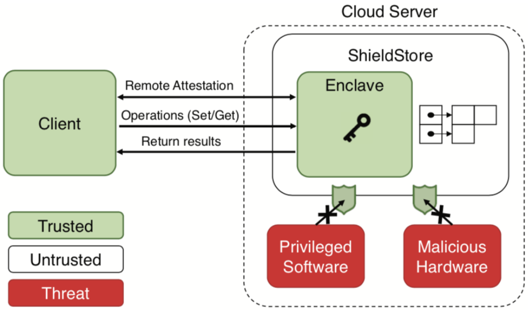


Figure 2.6: Overview of ShieldStore

First the client remote attests the server-side (1) verifying SGX support of the processor, the code, and other critical memory state of an enclave. In a second step, the client and the server exchange sessions keys (2) in order to establish a secure connection, using Intel SGX libraries to do so. Using this newly generated session key, the client sends a request for an operation (3). The server deciphers and verifies the request and accesses the Key-Value Store (4). Clients do not access the server-side ciphertexts neither need to know the encryption key used by the server to encrypt the values. The server will then decrypt the data from the storage, encrypted it again with the session key and reply to the client (5). All accesses to the KVS have integrity checks.

2.3.4 Discussion

Concluding, EnclaveDB (section 2.3.3.1) and Pesos (section 2.3.3.2) presents secure databases and objects storage systems respectively, using SGX, but EnclaveDB assumes that SGX supports large enclaves whose size is an order of several hundred GBs and Pesos restricts the size of data structure to the size of EPC. On the other hand, Speicher (section 2.3.3.3) and ShieldStore (section 2.3.3.4) proposes a store that alleviates the memory limitation of Intel SGX by storing encrypted data on untrusted memory regions. Speicher and ShieldStore have similar architectures, but the former is primarily design for persistence storage and the latter is focused on a fast in-memory key-value store.

We can now conclude that the clever partitioning of the application into trusted and untrusted parts is really important when programming with Intel SGX. It directly affects syscalls and I/O, performance and memory of the service.

The long goal of this thesis is to implement a system with characteristics from the databases present above. In terms of performance we plan to implement partial homomorphic encryption, so it allows to perform operations directly over in-memory encrypted data. This will be a challenge, as fully homomorphic encryption is not yet practical [25], so adaptations must be made, but performance increase are expected over the databases presented above, by not needing to decrypt the data in secure execution. Persistence will also be a requirement just like some of the databases presented.

For trusted execution with SGX, extensive research is needed to partition the application in the two necessary modes to circumvent persistence, performance and memory SGX limitations. It will also be researched and tested the ability to provide built-in replication and availability with SGX.

2.4 Related Work Balance and Critical Analysis

In the current days, computer scientists are always looking for a secure, fast and cheap environment to develop applications. As we know, it is not feasible to have all three of this elements working flawlessly without any compromises. Although, by combining in-memory key value stores, trusted remote execution environments and cloud providers, developers can now have a practical example of what would be to develop for a privacy enhanced secure system with reduce costs by using cloud providers and with better reassurances that the hardware and software that it's out of the control of the user will have a minimal impact on sensitive data and code of an application. By adding the performance benefits of an in-memory key value store and all of its technology, like built-in security, built-in replication and persistence we can in the best of our abilities today, combine the best of the three worlds without compromising too much on any of them.

In this thesis we will compare different kinds of approaches to implement a fast system with the assurance of a secure data flow that can easily be deployed into the cloud without fear of any components out of our control.

Chapter 3

3. Approach to Elaboration Phase

This Chapter aims at exemplifying how to do common stuff with LaTeX. We also show some stuff which is not that common! ;)

Please, use these examples as a starting point, but you should always consider using the *Big Oracle* (aka, Google, your best friend) to search for additional information or alternative ways for achieving similar results.

3.1 Refinement of Objectives and Contributions

Refinement of objectives and contributions

3.2 System Model Approach

System model approach

3.3 Planned Architecture and Implementation

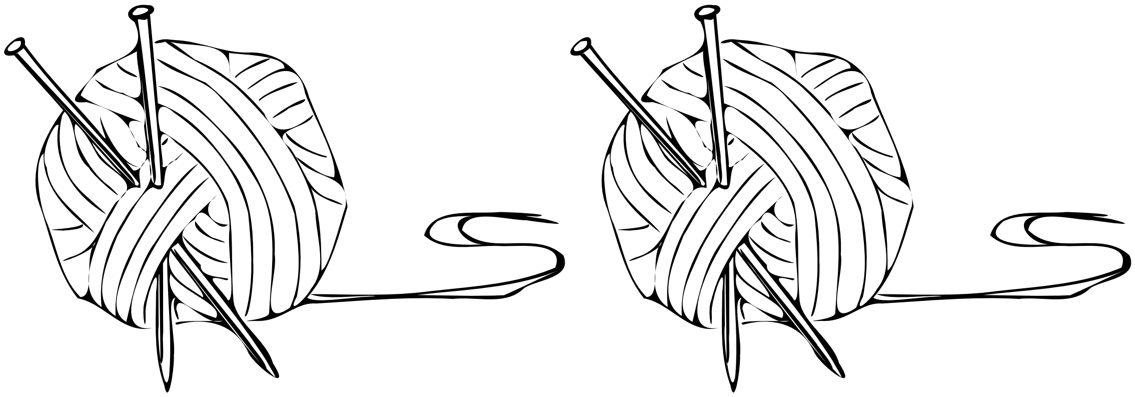
Planned architecture and implementation

3.4 Planned Testbench Environments

Planned testbench environments

3.5 Relevant Evaluation Criteria

Chapter 3



(a) One sub-figure

(b) Another sub-figure

Figure 3.1: A figure with two sub-figures!

And this is a small text that references the Figure 3.1 and its Subfigures 3.1a and 3.1b.

Chapter 3

Chapter 4

Workplan

Chapter 4

Bibliography

[1] *83% Of Enterprise Workloads Will Be In The Cloud By 2020*. Accessed: 2020-01-11. URL: https://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020/#2e513c5b6261.

[2] *A brief history of cloud computing*. Accessed: 2020-01-11. URL: https://www.ibm.com/blogs/cloud-computing/2014/03/18/a-brief-history-of-cloud-computing-3/.

[3] *A history of cloud computing*. Accessed: 2020-01-11. URL: https://www.computerweekly.com/feature/A-history-of-cloud-computing.

[4] *Aerospike*. Accessed: 2019-06-16. URL: https://www.aerospike.com.

[5] *Aerospike*. Accessed: 2019-06-16. URL: https://www.aerospike.com/docs/guide/security/index.html.

[6] *Amazon Dynamo DB*. Accessed: 2019-06-16. URL: aws.amazon.com/dynamodb.

[7] *Amazon S3 - Cloud Storage*. Accessed: 2019-07-09. URL: https://aws.amazon.com/s3/.

[8] *Amazon Web Services*. Accessed: 2019-06-16. URL: https://aws.amazon.com.

[9] *ARM TrustZone*. Accessed: 2020-01-08. URL: https://developer.arm.com/ip-products/security-ip/trustzone.

[10] *ARM TrustZone Security Whitepaper*. Tech. rep. PRD29-GENC-009492C. ARM Lmited, Dec. 2008. URL: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.

[11] *ARM TrustZone Stack Image*. Accessed: 2020-01-08. URL: https://malware.news/t/introduction-to-trusted-execution-environment-arms-trustzone/20823.

[12] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer. "SCONE: Secure Linux Containers with Intel SGX." In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI*

16). Savannah, GA: USENIX Association, 2016, pp. 689–703. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/arnautov.

[13] *Attestation and Trusted Computing - CSEP 590: Practical Aspects of Modern Cryptography*. Accessed: 2019-11-26. URL: https://courses.cs.washington.edu/courses/csep590/06wi/finalprojects/bare.pdf.

[14] *Attestation Identity Key (AIK) Certificate Enrollment Specification*. Accessed: 2019-11-26. URL: https://www.trustedcomputinggroup.org/wp-content/uploads/IWG-AIK-CMC-enrollment-FAQ.pdf.

[15] *AWS hit by major DDoS attack*. Accessed: 2020-01-11. URL: https://www.techradar.com/news/aws-hit-by-major-ddos-attack.

[16] *Azure Blob Storage*. Accessed: 2019-07-09. URL: https://azure.microsoft.com/en-us/services/storage/blobs/.

[17] M. Bailleu, J. Thalheim, P. Bhatotia, C. Fetzer, M. Honda, and K. Vaswani. "SPEICHER: Securing LSM-based Key-Value Stores using Shielded Execution." In: *17th USENIX Conference on File and Storage Technologies (FAST 19)*. Boston, MA: USENIX Association, 2019, pp. 173–190. ISBN: 978-1-931971-48-5. URL: https://www.usenix.org/conference/fast19/presentation/bailleu.

[18] S. Banescu. *Cache Timing Attacks*. July 2011.

[19] *Benefits of Hardware Trusted Modules*. Accessed: 2019-11-30. URL: https://www.hardware-security-module.com/benefits/.

[20] V. Costan and S. Devadas. *Intel SGX Explained*. Cryptology ePrint Archive, Report 2016/086. ’https://eprint.iacr.org/2016/086’. 2016.

[21] V. Costan, I. Lebedev, and S. Devadas. *Sanctum: Minimal Hardware Extensions for Strong Software Isolation*. Cryptology ePrint Archive, Report 2015/564. ’https://eprint.iacr.org/2015/
2015.

[22] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. "Dynamo: Amazon's Highly Available Key-value Store." In: *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*. SOSP ’07. Stevenson, Washington, USA: ACM, 2007, pp. 205–220. ISBN: 978-1-59593-591-5. DOI: 10.1145/1294261.1294281. URL: http://doi.acm.org/10.1145/1294261.1294281.

[23] *Document Store*. Accessed: 2019-06-16. URL: https://aws.amazon.com/nosql/document/.

[24] *Edward Snowden: Leaks that exposed US spy programme*. Accessed: 2020-01-11. URL: https://www.bbc.com/news/world-us-canada-23123964.

[25] C. Gentry, S. Halevi, and N. P. Smart. “Homomorphic Evaluation of the AES Circuit.” In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 850–867. DOI: 10.1007/978-3-642-32009-5_49. URL: https://doi.org/10.1007/978-3-642-32009-5_49.

[26] *Google Cloud Storage*. Accessed: 2019-07-09. URL: https://cloud.google.com/storage/.

[27] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller. “Cache Attacks on Intel 'SGX'.” In: *Proceedings of the 10th European Workshop on Systems Security - 'EuroSec' '17*. 'ACM' Press, 2017. DOI: 10.1145/3065913.3065915. URL: https://doi.org/10.1145/3065913.3065915.

[28] *Graph DBMS*. Accessed: 2019-06-16. URL: https://aws.amazon.com/nosql/graph/.

[29] *Hardware Trusted Modules*. Accessed: 2019-11-30. URL: https://resources.infosecinstitute.com/tpms-or-hsms-and-their-role-in-full-disk-encryption-fde/.

[30] *Intel SGX*. Accessed: 2020-01-04. URL: https://software.intel.com/en-us/sgx.

[31] S. IT. *Key-Value Stores*. Accessed: 2019-06-16. URL: https://db-engines.com/en/article/Key-value+Stores.

[32] S. IT. *Key-Value Stores Ranking*. Accessed: 2019-06-16. URL: https://db-engines.com/en/ranking/key-value+store.

[33] A. K. Khan and H. J. Mahanta. “Side channel attacks and their mitigation techniques.” In: *2014 First International Conference on Automation, Control, Energy and Systems ('ACES')*. IEEE, Feb. 2014. DOI: 10.1109/aces.2014.6807983. URL: https://doi.org/10.1109/aces.2014.6807983.

[34] T. Kim, J. Park, J. Woo, S. Jeon, and J. Huh. “ShieldStore: Shielded In-memory Key-value Storage with SGX.” In: *Proceedings of the Fourteenth EuroSys Conference 2019*. EuroSys '19. Dresden, Germany: ACM, 2019, 14:1–14:15. ISBN: 978-1-4503-6281-8. DOI: 10.1145/3302424.3303951. URL: http://doi.acm.org/10.1145/3302424.3303951.

[35] *Kinetic Object Storage*. Accessed: 2019-07-09. URL: https://storageioblog.com/seagate-kinetic-cloud-object-storage-io-platform/.

[36] R. Krahn, B. Trach, A. Vahldiek-Oberwagner, T. Knauth, P. Bhatotia, and C. Fetzer. “Pesos.” In: *Proceedings of the Thirteenth EuroSys Conference on - EuroSys 18*. ACM Press, 2018. DOI: 10.1145/3190508.3190518. URL: https://doi.org/10.1145/3190508.3190518.

[37] *Memcached*. Accessed: 2019-06-16. URL: http://www.memcached.org.

[38] *Memcached Github*. Accessed: 2019-06-16. URL: https://github.com/memcached/memcached/wiki/Overview.

[39] *Microsoft Azure Cache For Redis*. Accessed: 2019-06-16. URL: https://azure.microsoft.com/en-us/services/cache/.

[40] *Microsoft Azure Cosmos DB*. Accessed: 2019-06-16. URL: https://azure.microsoft.com/en-us/services/cosmos-db/.

[41] *Microsoft Azure Services*. Accessed: 2019-06-16. URL: https://azure.microsoft.com/en-us.

[42] *Microsoft Security Intelligence Report Volume 22*. Accessed: 2020-01-11. URL: https://www.microsoft.com/security/blog/2017/08/17/microsoft-security-intelligence-report-volume-22-is-now-available/.

[43] C. Priebe, K. Vaswani, and M. Costa. "EnclaveDB – A Secure Database using SGX." In: *To appear in the Proceedings of the IEEE Symposium on Security & Privacy, May 2018*. IEEE, May 2018. URL: https://www.microsoft.com/en-us/research/publication/enclavedb-a-secure-database-using-sgx/.

[44] *Professor John McCarthy*. Accessed: 2020-01-11. URL: https://cs.stanford.edu/memoriam/professor-john-mccarthy.

[45] *Redis*. Accessed: 2019-06-16. URL: https://redis.io.

[46] *Redis Multi Model Store*. Accessed: 2019-06-16. URL: https://db-engines.com/en/system/Redis.

[47] *Redis Performance Benchmark*. Accessed: 2019-06-16. URL: https://redislabs.com/docs/nosql-performance-benchmark/.

[48] *Redis Security*. Accessed: 2019-06-16. URL: https://redis.io/topics/security.

[49] *SASL Rfc*. Accessed: 2019-06-16. URL: https://tools.ietf.org/html/rfc2222.

[50] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard. "Malware Guard Extension: Using SGX to Conceal Cache Attacks." In: *arXiv e-prints*, arXiv:1702.08719 (Feb. 2017), arXiv:1702.08719. arXiv: 1702.08719 [cs.CR].

[51] *Search Engine Database*. Accessed: 2019-06-16. URL: https://aws.amazon.com/nosql/search/.

[52] *Sgx Memory Limits*. Accessed: 2019-06-16. URL: https://software.intel.com/en-us/forums/intel-software-guard-extensions-intel-sgx/topic/670322.

[53] *SGX Secure Enclaves in Practice: Security and Crypto Review*. Black Hat USA 2016 Security Conference. URL: https://www.youtube.com/watch?v=0ZVFy4Qsryc.

[54] *The Benefits Of Moving To The Cloud*. Accessed: 2020-01-11. URL: https://www.forbes.com/sites/forbestechcouncil/2017/05/19/the-benefits-of-moving-to-the-cloud.

[55] *Time Series Databases*. Accessed: 2019-06-16. URL: https://www.forbes.com/sites/metabrown/2018/03/31/get-the-basics-on-nosql-databases-time-series-databases/.

[56] *Trusted Platform Modules*. Accessed: 2019-07-09. URL: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/trusted-platform-module-overview.

[57] A. Waterman, Y. Lee, D. A. Patterson, K. Asanovic, V. I. U. level Isa, A. Waterman, Y. Lee, and D. Patterson. *The RISC-V Instruction Set Manual*. 2014.

[58] *What is a Trusted Execution Environment (TEE)?* Accessed: 2019-12-03. URL: https://www.trustonic.com/news/technology/what-is-a-trusted-execution-environment-tee/.

Appendix A

Appendix 1 Lorem Ipsum

Appendix 1

Appendix B

Appendix 2 Lorem Ipsum

Apendix 2

Annex I

Annex 1 Lorem Ipsum

Annex 1