Karlsruhe Institute of Technology

Confidential Computing in Distributed Systems

Bachelor’s Thesis of

Wenzhe Vincent Cui

at the Department of Informatics
SCC – Steinbuch Centre for Computing

Reviewer:	Prof. A
Second reviewer:	Prof. B
Advisor:	M.Sc. C
Second advisor:	M.Sc. D

11. Month 2021 – 02. Month 2022

I declare that I have developed and written the enclosed thesis completely by myself. I have submitted neither parts of nor the complete thesis as an examination elsewhere. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. This also applies to figures, sketches, images and similar depictions, as well as sources from the internet.

PLACE, DATE

. . . .

(Wenzhe Vincent Cui)

Todo list

Abstract

Distributed systems, like grid and cloud computing, have become essential for sharing of and accessing to remote resources. These systems are often used for economic reasons or to simplify collaboration and exchange of information. Because the maintenance and management of these distributed systems and the hardware it is running on is very complex and expensive, these tasks are often outsourced to service providers.

However, certain industries may have stricter security and trust requirements and governments too are growing more concerned with the privacy and confidentiality of user data. Confidential computing is an upcoming technology that seeks to address this trust issue by guaranteeing both privacy and integrity of data and code while enabling remote attestation.

This paper will explore different platforms that facilitate sharing of resources while make use of confidential computing in order to preserve the privacy of client data. It will also investigate limitations imposed by the use of confidential computing and finally showcase challenges of realizing a secure joint distributed system.

DSGVO?

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

1.2. Goal

This thesis focuses on two main goals:

- **I** Management of infrastructure and orchestration of application workloads running on this infrastructure. Provide application developers services that support them with the development of applications and the deployment of application workloads.
- **II** Make application workloads deployed by these services confidential (definition in section 2.1.2) and allow verification of this confidentiality by clients and third-parties.

Services mentioned in goal **I** include

- continuous integration (CI)
- continuous delivery and deployment (CD)
- deploying applications like services and computation tasks
- orchestration and monitoring of those applications

While these services should support the application owner in developing and deploying applications (**I.a**) it should impose as little limitations as possible on the application owner. Decisions regarding the programming language, libraries, and frameworks should be made by the application owner (**I.b**).

Goal **II** doesn't imply application security. Application workloads deployed with the help of the service provider should be shielded from the infrastructure, the service provider, and its services. But securing the application is outside the scope of this goal (**II.a**). The confidentiality should also not require manual application modification by the application owner (**II.b**).

1.3. Reference Use Cases

1.4. Research Methodology

2. Terminology

2.1. Terms

2.1.1. Platform

Group of technologies and services that are used as a base upon which applications are developed and deployed.

Expand explanation of term ‘Platform’

Go more into dynamic infrastructure management

2.1.2. Confidentiality

Isolation techniques like containerization and virtualization protect the infrastructure or platform from applications. On the other hand confidentiality is the direct opposite and its goal is to protect the application from the infrastructure or platform it is running on.

2.1.3. Trusted Computing Base (TCB)

The set of all hardware, firmware, and/or software components that are critical to the security of a given computing system. These components are the only components in the computing system that operate at a high level of trust. This doesn’t imply that these components are secure, but that they are crucial to the security of system as a whole.

2.2. Roles

This section will reference the background and cloud computing section below.

2.2.1. Service Provider

An entity providing services in order to ease the development and deployment of applications – often in the form of a platform.

2.2.2. Client

Entities that make use of the platform services provided by the service provider.

2.2.2.1. Application Owner

An entity developing an application on the platform provided by the service provider.

2.2.2.2. Data Owner

An entity that is in the possession of possibly sensible data that will be processed or used by an application deployed on the platform provided by the service provider.

2.2.2.3. Third Party

3. Background

3.1. Virtualization

3.1.1. Virtual Machine Manager (VMM)

3.2. Containerization

3.2.1. Container Manager (CM)

containerd, cri-o

3.2.2. Container Runtime (CR)

runc, kata

3.2.3. Shim

4. Technical Research

4.1. Cloud Computing

4.1.1. Definition

The NIST Definition of Cloud Computing [6]

4.1.1.1. Characteristics

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

4.1.1.2. Service Models

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

4.1.2. Benefits

4.1.3. Problems

4.2. Confidential Computing (CC)

Data can be in three distinct states: “at rest”, “in transit”, and “in use”. These three states describe data that lies in persistent storage, traversing the network, and data that is currently being processed. While technologies in order to protect data “at rest” and “in transit” are commonly used today, there are not many methods to protect data “in use”.

By executing computations in hardware-based trusted execution environments (see section 4.2.1) confidential computing (CC) protects data "in use". In order to assure a client that the requested environment can be trusted, remote attestation protocols (see section 4.3) are used. The combination of these two practices enable service providers to offer trusted services that can be verified remotely by clients or third-parties.

4.2.1. Trusted Execution Environments (TEEs)

4.2.1.1. Properties

There are different definitions of a trusted execution environment (TEE) with varying properties. We will focus on the properties needed to build a platform where securing client data and building trust is the main concern. These properties are:

Data confidentiality
Prevent unauthorized entities to view data that is in use within a TEE

Data integrity
Prevent unauthorized entities to add, remove, or change data while it is in use within a TEE

Code integrity
Prevent unauthorized entities to add, remove, or change code executing in the TEE

Explain why these three properties are important in order to build trust

4.2.1.2. Why hardware-based TEE

The security of a software layer can only be as strong as the layers below it. This is why an ideal security solution acts from the lowest layer possible. By providing security through the lowest layer – the hardware – it is possible to remove almost all layers below the TEE from its TCB. This includes the infrastructure provider, host operating system, hypervisor, service provider, and platform the TEE is running on. The only component remaining in the TCB of the application is the hardware providing the TEE properties.

Utilizing software-based TEEs would mean giving control of enforcing TEE properties in the hands of the service provider, conflicting with goal **II.a**. For this reason this paper will focus solely on hardware-based TEE technologies.

4.2.2. TEE Models

There are two distinct models of TEEs, process-based and VM-based.

4.2.2.1. Process-based TEEs

Process-based TEEs introduce a new programming model. A program needs to be split into two components, trusted and untrusted. These are often referred to as the "enclave" and "host". The enclave is executed in an environment where all TEE properties are provided. The enclave component should contain all code that interacts with sensitive data, whereas the host component is responsible for handling non-sensitive tasks like networking and file I/O.

While the host is not shielded the enclave is protected from the rest of the system, this includes

- the enclave's own host
- other processes
- the operating system
- the bootloader and firmware such as the BIOS
- the hypervisor and host operating system (in virtualized environments)
- hardware other than the processor

Go into detail about how process-based TEEs work?

Splitting a program into enclave and host is challenging. It requires a deep understanding of security and how these process-based TEE solutions work. To ease the development of such applications SDKs and frameworks often hide the split between host and enclave from the developer[7].

Library OSes like Gramine and Occlum go even further and provide a POSIX-like runtime environment with support for network, file I/O, and multithreading. These library OSes contain the whole application in the enclave. But because the enclave doesn't have access to the OS running the program, the library OS provides libraries which implements OS system calls as library functions. This library then interfaces with a boilerplate host for I/O[8].

Even though these SKDs, frameworks, and library OSes ease the development and make porting of existing applications easier, using process-based TEEs still requires more development effort and modifications of existing applications.

Example: Intel SGX

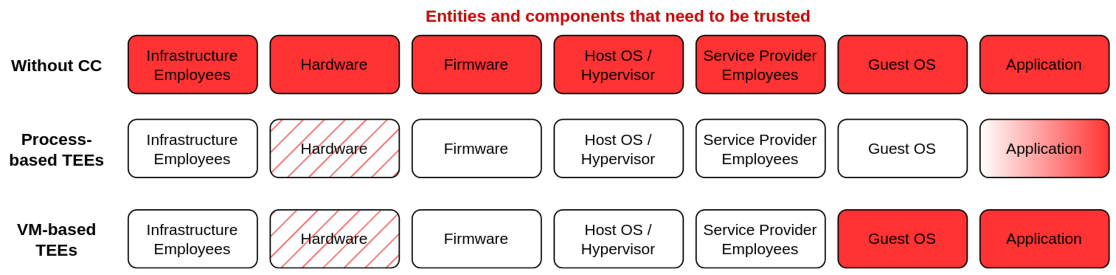

Figure 4.1.: Comparison of TEE models. Boxes marked in red highlight entities and components that have to be trusted. In both TEE models the CPU still enforces the TEE properties and thus still has to be trusted. In the process-based TEE the application is marked with a gradient because the application has to be split into two parts.

4.2.2.2. VM-based TEEs

The main concept of VM-based TEEs is to apply the TEE properties to a full virtual machine running on top of a VMM. This enables VM-based TEEs to run basically any application without modifications. While VM-based TEEs have a larger TCB than process-based TEEs they are explicitly designed to protect workloads from underlying software layers.

Example: AMD SEV

Example: Intel TDX

4.2.2.3. Comparison

Figure 4.1 shows a simplified comparison between the TCBs of an application running without CC, inside a process-based TEE, and inside a VM-based TEE. The main difference is the size of the TCB. While process-based TEEs should be used for security critical applications, the need to modify existing applications in order to run them conflicts with goal **II.b**.

But VM-based TEEs align with goal **II.a** by shielding the virtual machine and thereby applications running inside it from the outside, while also allowing the execution of unmodified applications which doesn't conflict with goal **II.b**.

4.2.3. Problems

4.2.3.1. Performance impact

Find performance comparisons

4.2.3.2. Support for accelerators

Reference the heavy use of accelerators in ML

Explain ongoing effort without going too much into detail (Reference CRONUS)

4.3. Remote Attestation

Running applications inside a CC enabled environment is not enough in order to deepen the trust with clients or third-parties. In order to prove that the application is running inside a CC enabled environment and that neither the application nor the system it is running on has been tampered with, the service provider can facilitate remote attestation.

In a remote attestation environment there are at least two roles: the attester and the relying party. The attester produces information about itself (evidence), on which the relying party makes a decision whether to trust the attester or not. In our context the attester would be the system running a client application and the relying party would be the client accessing or sending data to the application.

4.3.1. Remote Attestation Procedures (RATS)

Remote Attestation Procedures (RATS) defines a general architecture, roles, and messages in order to establish trust between the relying party and the attester. Figure 4.2 shows the general architecture. RATS introduces few new roles where the most prominent one is the verifier. It produces attestation results by using evidence, endorsements, reference values, and applying an appraisal policy to assess the trustworthiness of the attester. These attestation results then support the decision process of the relying party on whether to trust the attester or not. We will go into more detail in chapter 5.1.

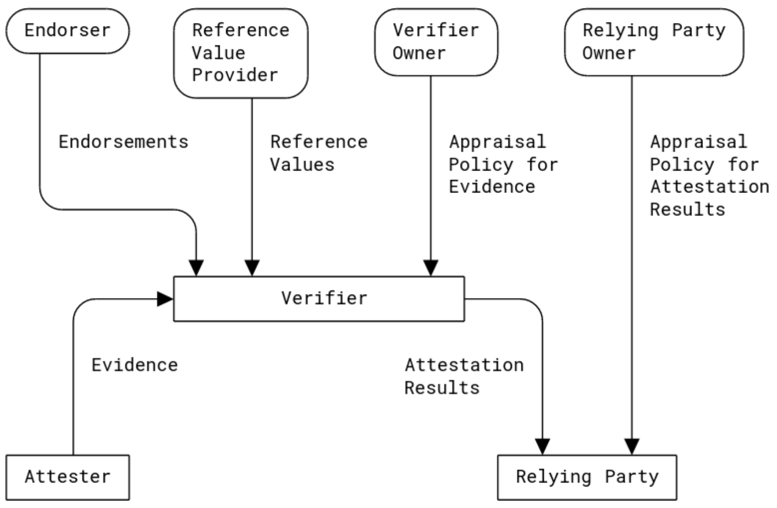

Figure 4.2.: RATS architectural overview.
Source: *Remote ATtestation procedureS (RATS) Architecture* [3]

4.4. Existing Solutions

4.4.1. Commercial Solutions by well-known Service Providers

AWS Nitro

Description: AWS Nitro

- More of a TEE technology
- Doesn't solve trust problem with the service provider

Confidential VMs (Azure, GCP)

Description: Azure Confidential Containers

-

GCP Confidential Dataproc

Big data processing through fully managed data processing frameworks and tools like Spark and Hadoop. Uses confidential VMs to provide confidentiality.

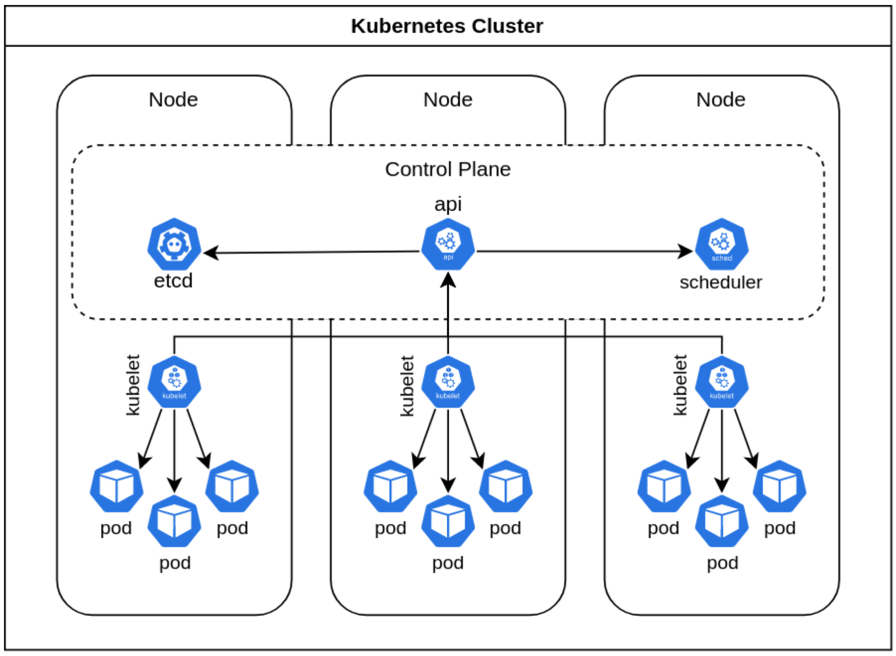

Figure 4.3.: A simplified overview over the Kubernetes components.

GCP Confidential Space
Let multiple parties share confidential data with a workload while retaining the confidentiality and ownership of the data.

Problems with commercial solutions

4.4.2. Kubernetes as platform base

Kubernetes is an extensible, open source platform for managing containerized workloads and services. Due to its open source nature and popularity there has been a rapidly growing ecosystem surrounding Kubernetes. It provides general platform features such as deployment, orchestration, scaling, load-balancing, integration with logging, monitoring, and alerting solutions.

A simplified overview over core components inside a Kubernetes cluster (see figure 4.3):

Nodes
(Possibly virtual) machines running containerized applications. On each node

runs a component called kubelet that manages the pods and the corresponding containers on the node.

Pods A group of containers that share storage and network resources that models an application-specific logical host. The containers that make up a pod are always located on the same node and are scheduled in unison.

Control Plane
A collection of pods that manage the nodes and workload pods inside the cluster. This includes an API, a backing store (etcd) for all cluster data, and a scheduler that assigns pods to nodes.

Container Runtime
Responsible for executing containers on a node (see section 3.2.2).

Kubernetes: Expand as needed in the next chapters

Kubernetes' non-monolithic design allows the replacement of almost every aspect and component of a cluster. This allows a Kubernetes cluster to be run on basically any underlying infrastructure, regardless of the hardware choices or networking design.

There are three levels in a Kubernetes cluster on which confidentiality can be applied to: The node, pod, or container. There are distinct benefits and problems for applying confidentiality on each layer. We will discuss them in the following sections.

4.4.2.1. Confidential Nodes

The most outer layer where confidentiality can be applied to in the Kubernetes architecture are the nodes. By using confidential computing enabled virtual machines – for example by facilitating AMD SEV or Intel TDX – a Kubernetes cluster operator is able to shield workloads running inside the cluster. Prominent service providers like Azure and GCP already offer the option of deploying their managed Kubernetes clusters with confidential worker nodes. However, these solutions don't include verification of the nodes which means that one would have to build a custom remote attestation system – for example by implementing the RATS (section 4.3.1).

The biggest problem with confidential nodes is the restriction of cluster admin privileges and verifying these as a client. While the workloads are shielded from the infrastructure cluster administrators would still have full control over the workloads running inside the cluster which breaks goal **II.a**.

4.4.2.2. Confidential Containers

The other extreme would be to apply confidentiality to containers. A process running inside a TEE would only receive confidential data like decryption keys or personalized information after verifying that the process is running inside a TEE via remote attestation. Since the TEE shields the process from the cluster this approach remove the cluster administrator from the TCB of the application.

Even though this approach doesn't conflict with the goals of this paper, it does collide with the design of Kubernetes. In the Kubernetes architecture the smallest deployable unit of computation is a pod, a collection of application containers. These containers share storage and network resources, but it becomes very hard to share these resources when the containers are shielded from each other. The next approach addresses this architectural issue.

4.4.2.3. Confidential Pods

Instead of applying confidentiality to containers we can also shield pods from the outside world. This improves the last approach as per definition a pod in a Kubernetes cluster defines an application-specific logical host. As opposed to shielding a single container, shielding a pod would allow sharing storage and networking resources between containers composing the pod.

5. Secure Computation Platform

5.1. Architecture

5.1.1. Design Choices

Containers

- Puts control over development environment into the hands of application owner
- No complex language support and dependency management by the service provider
- No language/dependency limitations

PaaS Service Model

Alignment with goal

Kubernetes

Kubernetes: Helps in achieving goal **I**

5.2. Implementation Challenges

	Infrastructure as a Service	Platform as a Service	
		Container	Binary
Provides	VMs with CC enabled	Container runtime that runs containers in a CC enabled VM	CC enabled runtime environment
Infrastructure & Orchestration managed by	Application Owner	Service Provider	Service Provider
Development Environment managed by	Application Owner	Application Owner	Service Provider
Notes	• Everything has to be managed by the application owner • Goal is to reduce complexity	• Big attack surface (TCB) if done bad • Much more in control of the service provider • Services supporting CC usage can be provided by the service provider	• Least control by application owner • Either complex dependency management and language support by the service provider • Or language or dependency limitations

Table 5.1.: An overview over different services models.

5.3. Evaluation

6. Conclusion

…

Bibliography

[1] *A Technical Analysis of Confidential Computing*. URL: https://confidentialcomputing io/ccc-a-technical-analysis-of-confidential-computing-v1-3_updated_november_2022/.

[2] Raad Bahmani et al. "{CURE}: A Security Architecture with {CUstomizable} and Resilient Enclaves". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 1073–1090.

[3] Henk Birkholz et al. *Remote ATtestation procedureS (RATS) Architecture*. RFC 9334. Jan. 2023. DOI: 10.17487/RFC9334. URL: https://www.rfc-editor.org/info/rfc9334.

[4] *Building a Secure System using TrustZone Technology*. 2009.

[5] Jianyu Jiang et al. "CRONUS: Fault-isolated, Secure and High-performance Heterogeneous Computing for Trusted Execution Environment". In: *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2022, pp. 124–143. DOI: 10.1109/MICRO56248.2022.00019.

[6] Tim Grance Peter Mell. *The NIST Definition of Cloud Computing*. URL: https://doi.org/10.6028/NIST.SP.800-145.

[7] Felix Schuster. *Confidential Computing - How to process data securely on third-party infrastructure*. URL: https://content.edgeless.systems/hubfs/Confidential%20Computing%20Whitepaper.pdf.

[8] Chia-Che Tsai et al. "Cooperation and security isolation of library OSes for multi-process applications". In: *Proceedings of the Ninth European Conference on Computer Systems*. 2014, pp. 1–14.

[9] Jianping Zhu et al. "Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment". In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 1450–1465. DOI: 10.1109/SP40000.2020.00054.

A. Appendix

A.1. First Appendix Section

Figure A.1.: A figure

...