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)

# 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.

# 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 **??**) and allow verification of this confidentiality by clients and third-parties.

Services mentioned in goal **I** could include

- 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** does not 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**).

# 2. Secure Computation Platform Proposal

## 2.1. Trust Model

In the following chapter we will often refer to entities and components as “trusted” and “untrusted”. This trust relation is defined from the perspective of the data owner as this role is in possession of the sensitive data.

### 2.1.1. Roles

#### Infrastructure and Service Provider

The infrastructure provider is responsible for the availability of the infrastructure used to provide compute, networking, and storage resources. As such the infrastructure controls and has access to the physical hardware and firmware.

On the other hand the service provider is responsible for managing services that utilize the infrastructure provided by the infrastructure provider in order to ease the development and deployment of workloads. As such the service provider is not only responsible for deploying the applications, but also to provide services that allow the retrieval of attestation evidence and initialize TEE environments.

As stated in the introduction the goal is to remove these two entities from the trust model of the application. For this reason both roles can be aggregated in the same entity.

#### Application Owner

The application owner designs and implements application workloads that will be deployed and orchestrated by the service provider. This role needs to prove to the customer aspects of compliance to the defined requirements. It also defines computing resource requirements for the application workloads in order to run and maintain compliance.

#### Data Owner

As the owner of the confidential data, this role is concerned with the visibility and integrity of their data and the compliance of the application with the requested requirements.

#### Verifier

In order to remove the service provider from the application’s trust model, an external party has to verify the service provider’s confidentiality claims. The verifier provides an attestation service that appraises evidence and returns attestation results.

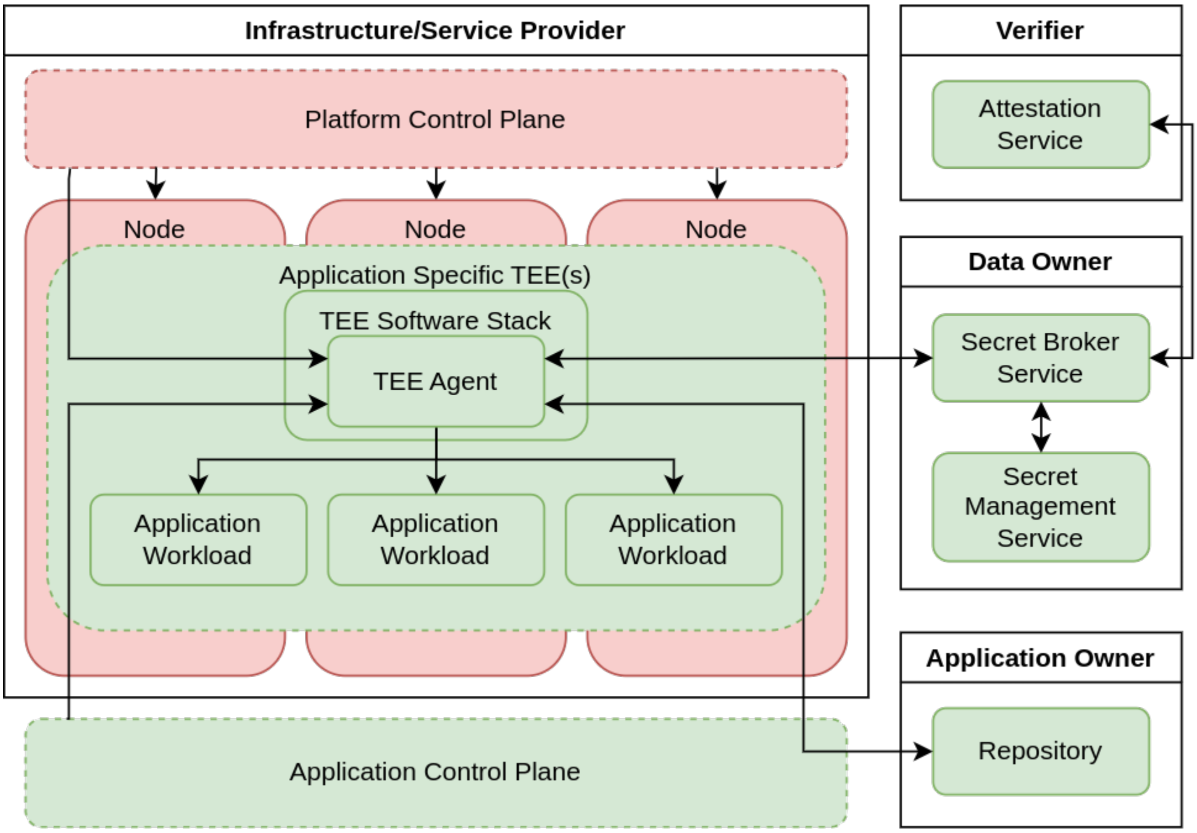

Figure 2.1.: An overview over components used in this architecture proposal. Defines which components are managed by which role and whether it is trusted. Entities and Components marked in red are untrusted, while those marked in green are trusted.

## 2.2. Architecture

### 2.2.1. Overview

For convenience, from now on we will refer to confidential data as “secrets”. Figure 2.1 shows an overview over the components used in this architecture proposal, aiding the reader with understanding the relation between the components introduced in the next section.

### 2.2.2. Components

**Control Planes**

In a dynamically provisioned platform managing the infrastructure and orchestrating applications is the responsibility of the control plane. We are grossly simplifying the control plane here by considering it as a single component. In traditional platforms the control plane is a high-privileged component in order to perform management and orchestration tasks.

We want to apply the least privileges principle to the control plane and only allow privileges necessary for basic management and orchestration. This includes managing the lifecycle of applications on a node, migration of applications between nodes, and monitor computing resource usage. But privileges that have the potential to compromise the integrity of application code and data should only be assigned to trusted entities and components. This implies that entities and components controlled by the service provider should not have these privileges.

To achieve this the traditional control plane has to be split into a platform and an application control plane. As stated above the platform control plane manages and orchestrates the platform with only the set of permissions that is needed to achieve this. On the other hand the application control plane is responsible for everything that directly interacts with the application. To make the split of the control plane effective the service provider can not control or have access to the application control plane. This implies that an external entity like the application or data owner has to manage this control plane instance.

#### TEE Agent

TEEs are designed to shield everything within the environment from the outside. Because the control plane is not part of a specific application TEE, the control plane is not able to interact with applications inside a TEE. To allow and control interactions with between the control plane and applications a new component is needed. The TEE agent itself is executed inside a TEE – not necessarily the same TEE as the application – and is responsible for managing the application specific workloads. It controls the access into the application TEE in order to perform management and orchestration actions and is responsible for enforcing the split control planes' permissions.

In a more traditional platform each node inside a platform cluster runs an agent which manages the applications running on the particular node. But because the node is under the control of the infrastructure or service provider the node and everything running on it can not be trusted. This is why an application specific agent is needed.

The TEE agent is the trusted component that is responsible for the whole attestation workflow. It is responsible for pulling application code, verifying the integrity of the code, executing application code inside a TEE, and injecting secrets into the application TEE. As such the enclave agent itself has to be verified before sending secrets to it. In section 2.3 we will go more into detail how this verification of the TEE agent and application code is performed.

#### Repository

In order to mitigate supply chain attacks the application owner has to store application code in a trusted signed repository or sign the application code itself. This signature then has to be validated by the TEE agent before executing the code on confidential data.

#### Secret Management Service

The secret management service is a component that stores confidential data and controls the access this data. This management of this service has to be done by the data owner or delegated to a trusted entity.

#### Secret Broker Service

The secrets broker service receives secrets requests from the TEE agent with evidence about its own integrity. This evidence is then forwarded to the attestation service which returns attestation results on which the secrets broker service then decides whether to trust the TEE agent. If the TEE agent is trusted the secret broker service then relays the secrets from the secret management service to the TEE agent.

#### Attestation Service

In the RATS architecture the attestation service takes on the role of the verifier. It appraises evidence from the TEE agent and returns attestation results to the secret broker service. See section **??** for more information about how the appraisal of evidence is performed.

## 2.3. Attestation Workflow

Following is the workflow when the deployment of an application is requested:

1. The platform control plane spawns a new TEE on an arbitrary node.
2. After initialization of the TEE the whole TEE software stack is measured.
3. The measurements are then sent as evidence to the secret broker service.
4. The secret broker service lets the attestation appraise the evidence and then decides whether to trust the TEE agent.
5. If the agent is trusted the secret broker service queries needed secrets from the secret management service and sends them to the TEE agent. These secrets have to include repository or application code signatures.
6. After receiving the secrets the TEE downloads application code from the repository and verifies it with the signatures it received.
7. The TEE agent then starts the application workloads in a TEE. This could be the same TEE where the agent is running or a new workload specific TEE.

## 2.4. Comparison to traditional architecture

## 2.5. Evaluation

# 3. 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] 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`.

[5] Claus Pahl. "Containerization and the PaaS Cloud". In: *IEEE Cloud Computing* 2.3 (2015), pp. 24–31. DOI: `10.1109/MCC.2015.51`.

[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.

# A. Appendix

## A.1. First Appendix Section

Figure A.1.: A figure

…