Teacleave: A Universal Secure Computing Platform

Mingshen Sun
Baidu, Apache Teacleave (incubating) PPMC

2nd SGX Community Workshop (July 14, 2020)
• Today we can build SGX application with SDK

• or we can deploy legacy application in containerized SGX environment based on LibOS and Unikernel concepts

• Still, lots of effort for developers
We need a **framework or platform** that allow the programmer to **concentrate on the business logic** and automates more protection of their code and data without worrying about technical details of different TEE implementations.
Teaclave

• Apache Teaclave (incubating) is an open source universal secure computing platform, making computation on privacy-sensitive data safe and simple.
  
  • Originally developed at Baidu called MesaTEE/Rust SGX SDK, open-source in July 2019
  
  • Entered Apache Incubator on August 2019, using Teaclave as the project name
  
  • Open source in The Apache Way
  
  • Homepage: https://teaclave.apache.org/
  
  • Repository
    • https://github.com/apache/incubator-teaclave
    • https://github.com/apache/incubator-teaclave-sgx-sdk
Highlights

• Functionality
  • function-as-a-service interfaces
  • built-in functions and Python executors

• Security
  • Intel SGX: hardware-based isolation, memory encryption and attestation
  • Rust: fast, memory-safe, system programming language

• Usability
  • deployment on the cloud infrastructure
  • API, SDK, CLI, SGX tool, etc

• Modularity
  • attestation, RPC, functions, binder
Workflow

• FaaS interface
  • function: business logic
  • data: sensitive data
  • participants: parties involved in a task

• Workflow of a task in Teaclave
  1. register sensitive data into the platform
  2. register a function you want to execute with the data
  3. create a task
  4. run the task and get results
Teaclave Design

Clients

Authentication Service

Frontend Service

Management Service

Scheduler Service

Access Control Service

Execution Service

api endpoint connections

internal endpoint connections
Domains
Service Overview

Clients

Authentication Service

Frontend Service

Management Service

Scheduler Service

Execution Service

Json Web Token (JWT) as our reference implementation

LevelDB

Stateless service, designed to be deployed in cloud infra

Management function/data registration, task approve/invocation, access control, data persistency, etc.

An access control engine using Python-like DSL

Authenticate requests, forward to management service and can provide potential load balancing

Dispatch tasks to workers with desired capabilities

api endpoint connections

internal endpoint connections

Flexible, can support all kinds of SGX runtime and other TEE.
Clients authenticate their id/credential to get a session key. This session key will be used later to communicate with the frontend service.

The frontend service will authenticate all client requests. Redirect valid requests to the management service.

Persist functions, data, task into database.

Fetch task ready for execution from database.

Subscribe/Pull tasks, then execute functions. Update status and execution results.
Clients authenticate their id/credential to get a session key. This session key will be used later to communicate with the frontend service.

The frontend service will authenticate all client requests.

Persist functions, data, task into database.

Fetch task ready for execution from database.

Subscribe/Pull tasks, then execute functions. Update status and execution results.

Persist functions, data, task into database.

Get Authorization of data/function usages, task invocation, etc.

Redirect valid requests to the management service.

Client then registers data and function if needed, and assign, approve, invoke task. Clients can also get information of functions, task, etc.

Service Interfaces Overview
Frontend Service

Authentication Service

Management Service

Storage Service

Scheduler Service

Access Control Service

Clients

RPC /rpc

service impl /services

function impl /worker

app/enclave communication /binder

service protocol def /services/proto

attestation /attestation, /rpc, /dcap

build/runtime configuration /config

Others

• /types
• /tests
• /examples
• /sdk
• /file_agent
• /common
• /utils
• /docker
• /cli
• /crypto
• /tool

Mics

• /third_party
• /licenses
• /docs
• /keys

API endpoint connections

Internal endpoint connections

Codebase Overview
Attestation

• Attestation in TLS handshake

Client

Server Cert Verifier

ClientHello

Server Hello

Server Certificate
Server Key Exchange
Server Hello Done

Client Certificate
Client Key Exchange
Change Cipher Spec
Finished

Change Cipher Spec
Finished

Application Data

Server

Client Cert Verifier
Attestation

• Server certificate verifier

- verify report with AS's public key
- verify mr_enclave with enclave_info.toml
- verify other items in report to decide whether to trust this server
Attestation

- Server certificate verifier

Server Certificate (X509)

Extension attestation report

Key...

- verify report with AS's public key
- verify mr_enclave with enclave_info.toml
- verify other items in report to decide whether to trust this server

enclave_info.toml is generated at build time containing information like mr_signer and mr_enclave of all enclaves.

enclave_info.toml should be signed by all auditors and will be verify at the startup of a service.
Getting Started

• Try
  • My First Function
  • Write Functions in Python
  • How to Add Built-in Functions

• Design
  • Threat Model
  • Mutual Attestation: Why and How
  • Access Control
  • Build System
  • Teaclove Service Internals

• Contribute
  • Rust Development Guideline
  • Development Tips

Documentation
https://teaclove.apache.org/docs/
Teaclace Community

Platform Users
Service Users
Library Users
SGX SDK Users

Commercial Products
Academic Research
Open Source Projects
Teaclave Community

Organizations

- Baidu
- ByteDance
- Enigma
- LayerX

Projects

- **Advanca**: A privacy-preserving general-purpose compute/storage infrastructure for Dapps.
- **Anonify**: A blockchain-agnostic execution environment with privacy and auditability based on TEE.
- **Enigma Core**: Enigma Core library. The domain: Trusted and Untrusted App in Rust.
- **Crypto.com Chain**: Alpha version prototype of Crypto.com Chain.
- **Occlum**: Occlum is a memory-safe, multi-process library OS for Intel SGX.
- **Phala Network**: A TEE-Blockchain hybrid architecture implementing Confidential Contract on Polkadot.
- **Secret Network**: A blockchain-based, open-source protocol that lets anyone perform computations on encrypted data, bringing privacy to smart contracts and public blockchains.
- **substraTEE**: Trusted Off-Chain Compute Framework for substrate blockchains.

https://teaclave.apache.org/community/
Thank you!

• Join us on our mailing list: https://lists.apache.org/list.html?dev@teaclave.apache.org

• Visit our homepage: https://teaclave.apache.org/

• Follow us at @ApacheTeaclave

• Checkout our code: https://github.com/apache/incubator-teaclave

• Contributors: https://teaclave.apache.org/contributors/

• Call for collaborations and contributors!