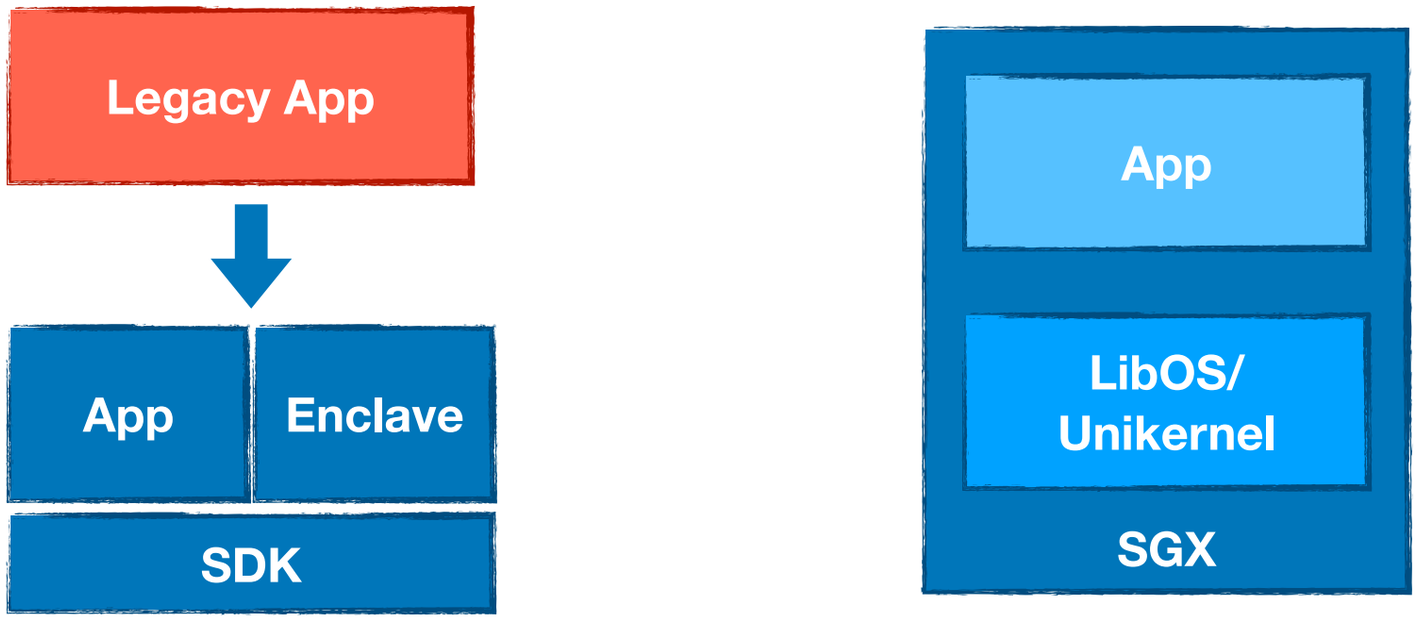
Teaclave: A Universal Secure Computing Platform

Mingshen Sun

Baidu, Apache Teaclave (incubating) PPMC

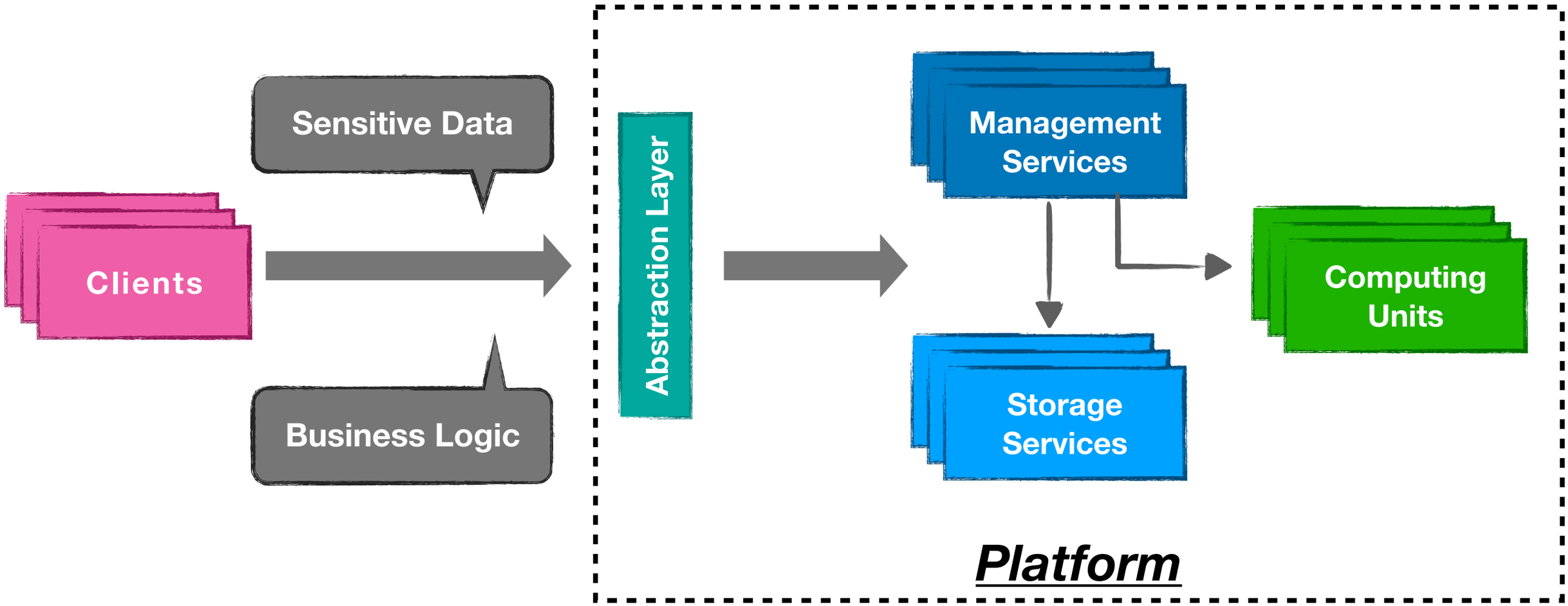
2nd SGX Community Workshop (July 14, 2020)

SGX Ecosystem: Now and Next



- 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

SGX Ecosystem: Now and Next



- 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

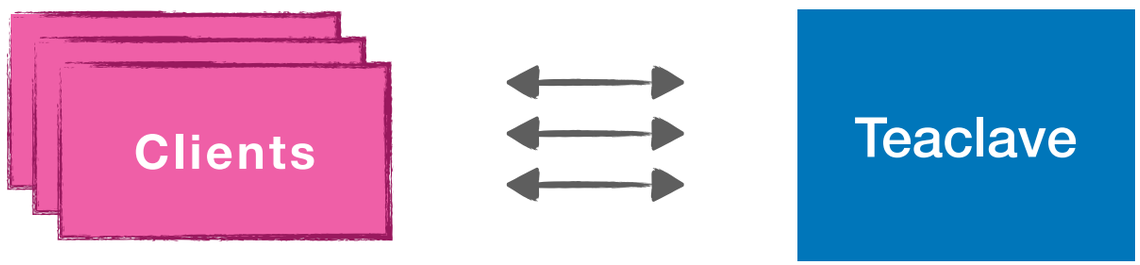
Highlights

- **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

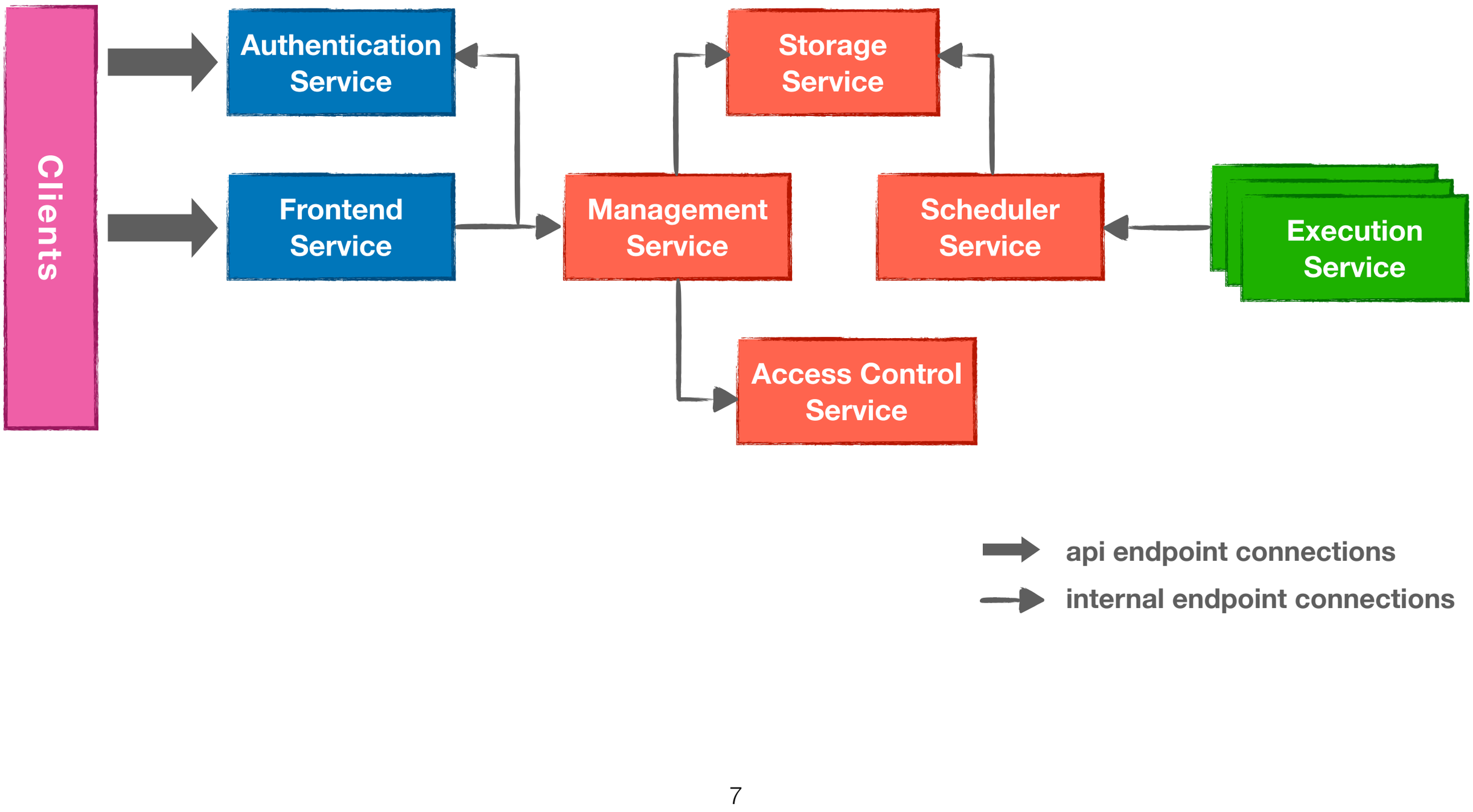
Workflow

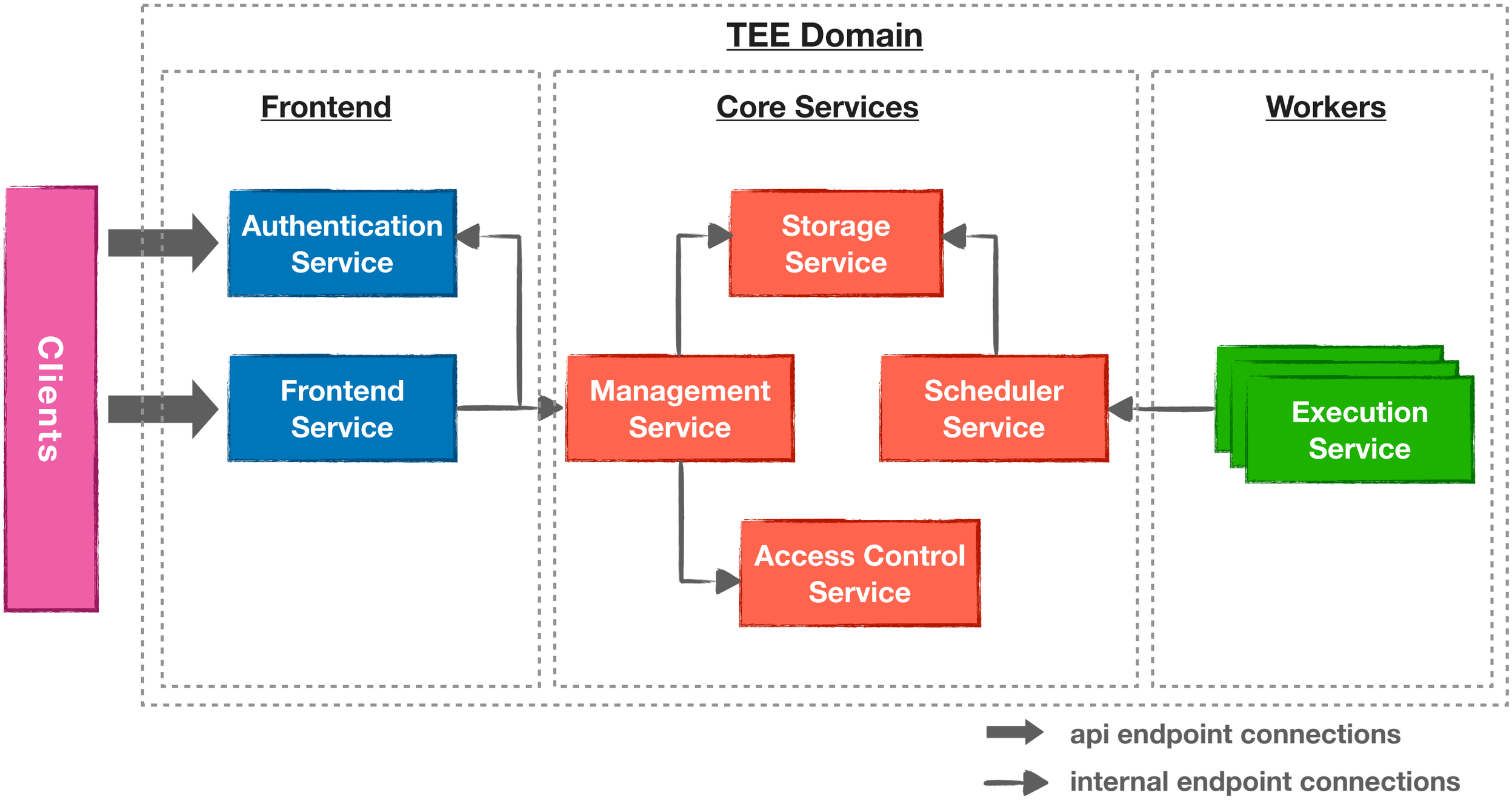
- **FaaS interface**
  - function: business logic
  - data: sensitive data
  - participants: parties involved in a task
- **Workflow of a task in Teaclave**

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



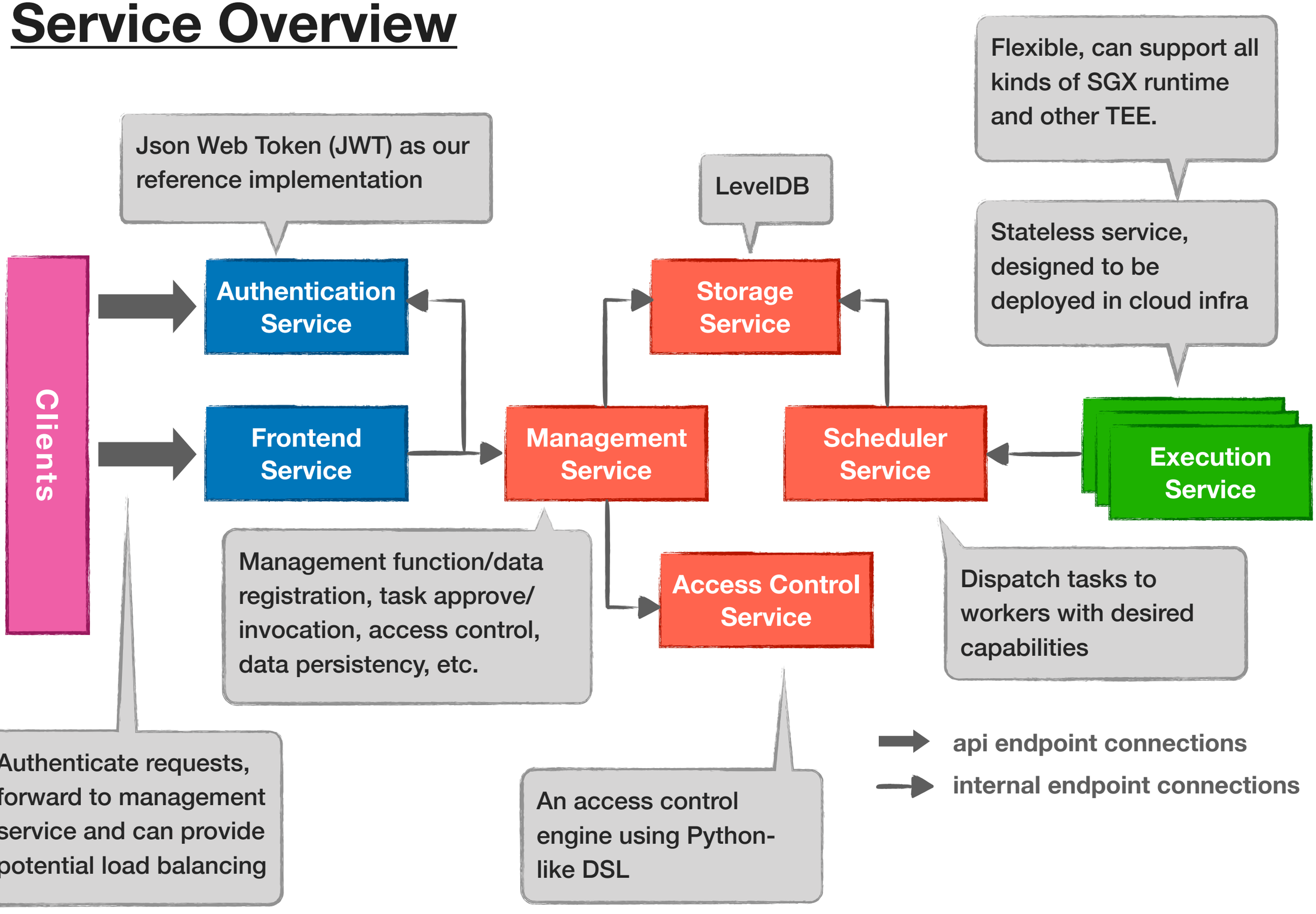
Teaclave Design




TEE Domain
Frontend
Core Services
Workers
Clients
Authentication Service
Frontend Service
Storage Service
Management Service
Scheduler Service
Execution Service
Access Control Service
api endpoint connections
internal endpoint connections

Domains

Service Overview



RPC Interfaces

```
service TeaclaveStorage {
  rpc Get
  rpc Put
  rpc Delete
  rpc Enqueue
  rpc Dequeue
}
```

```
service TeaclaveAuthenticationApi {
  rpc UserRegister
  rpc UserLogin
}

service TeaclaveAuthenticationInternal {
  rpc UserAuthenticate
}
```

```
service TeaclaveScheduler {
  rpc PublishTask
  rpc Subscribe
  rpc PullTask
  rpc UpdateTaskStatus
  rpc UpdateTaskResult
}
```

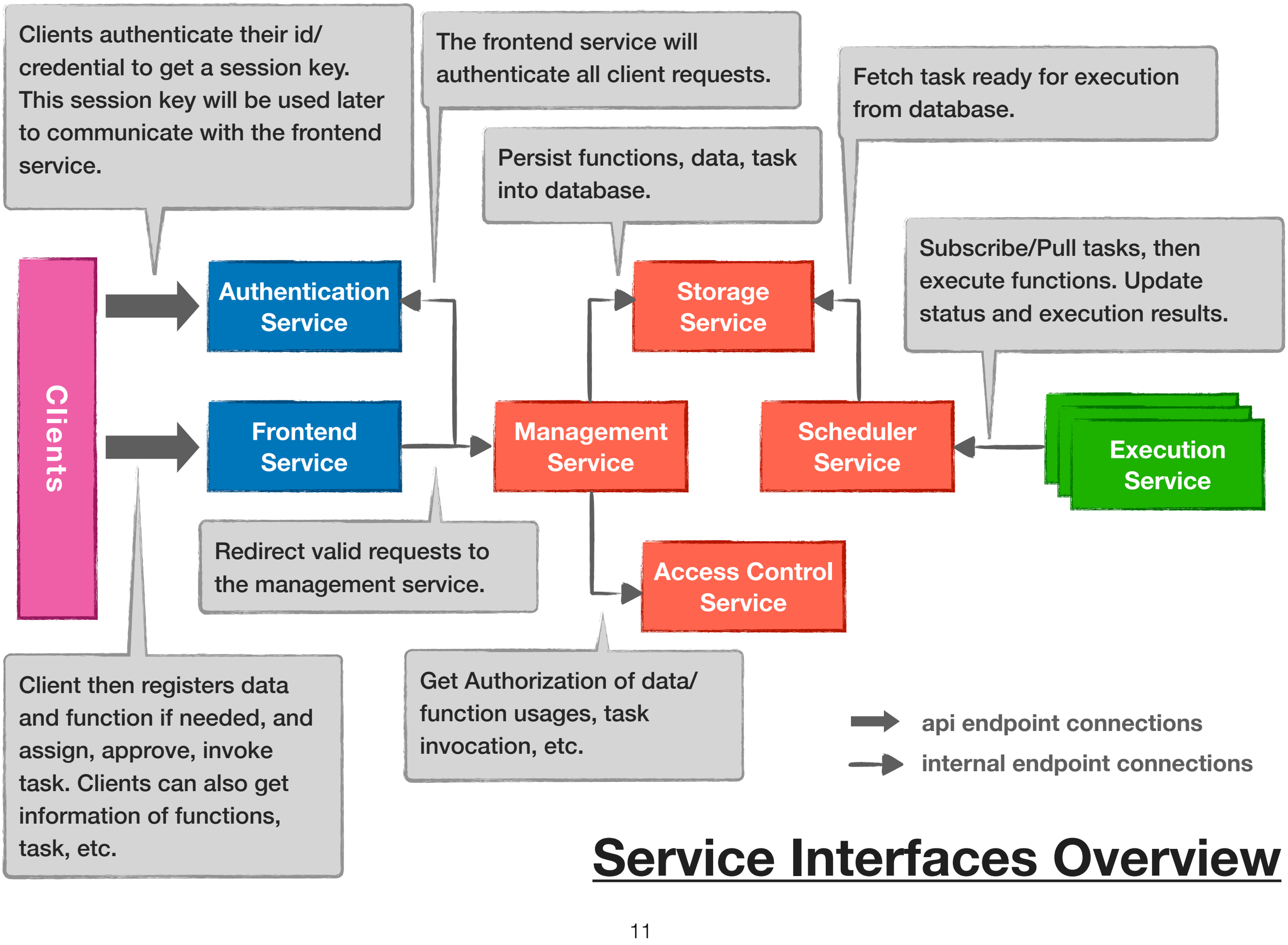
```
service TeaclaveManagement {
  rpc RegisterInputFile
  rpc RegisterOutputFile
  rpc RegisterFusionOutput
  rpc RegisterInputFromOutput
  rpc GetOutputFile
  rpc GetInputFile
  rpc RegisterFunction
  rpc GetFunction
  rpc CreateTask
  rpc GetTask
  rpc AssignData
  rpc ApproveTask
  rpc InvokeTask
}
```

```
service TeaclaveExecution {
}
```

```
service TeaclaveFrontend {
  rpc RegisterInputFile
  rpc RegisterOutputFile
  rpc RegisterFusionOutput
  rpc RegisterInputFromOutput
  rpc GetOutputFile
  rpc GetInputFile
  rpc RegisterFunction
  rpc GetFunction
  rpc CreateTask
  rpc GetTask
  rpc AssignData
  rpc ApproveTask
  rpc InvokeTask
}
```

```
service TeaclaveAccessControl {
  rpc AuthorizeData
  rpc AuthorizeFunction
  rpc AuthorizeTask
  rpc AuthorizeStagedTask
}
```

Service Interfaces Overview



Codebase Overview

RPC
`/rpc`

service impl
`/services`

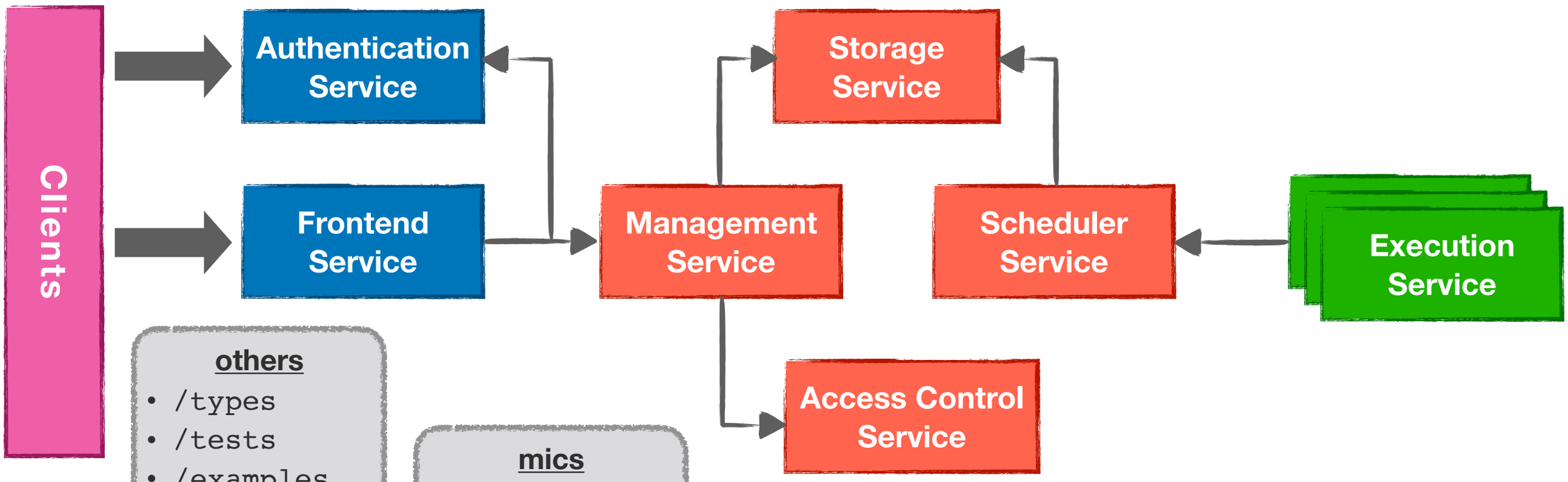
function impl
`/worker`

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`

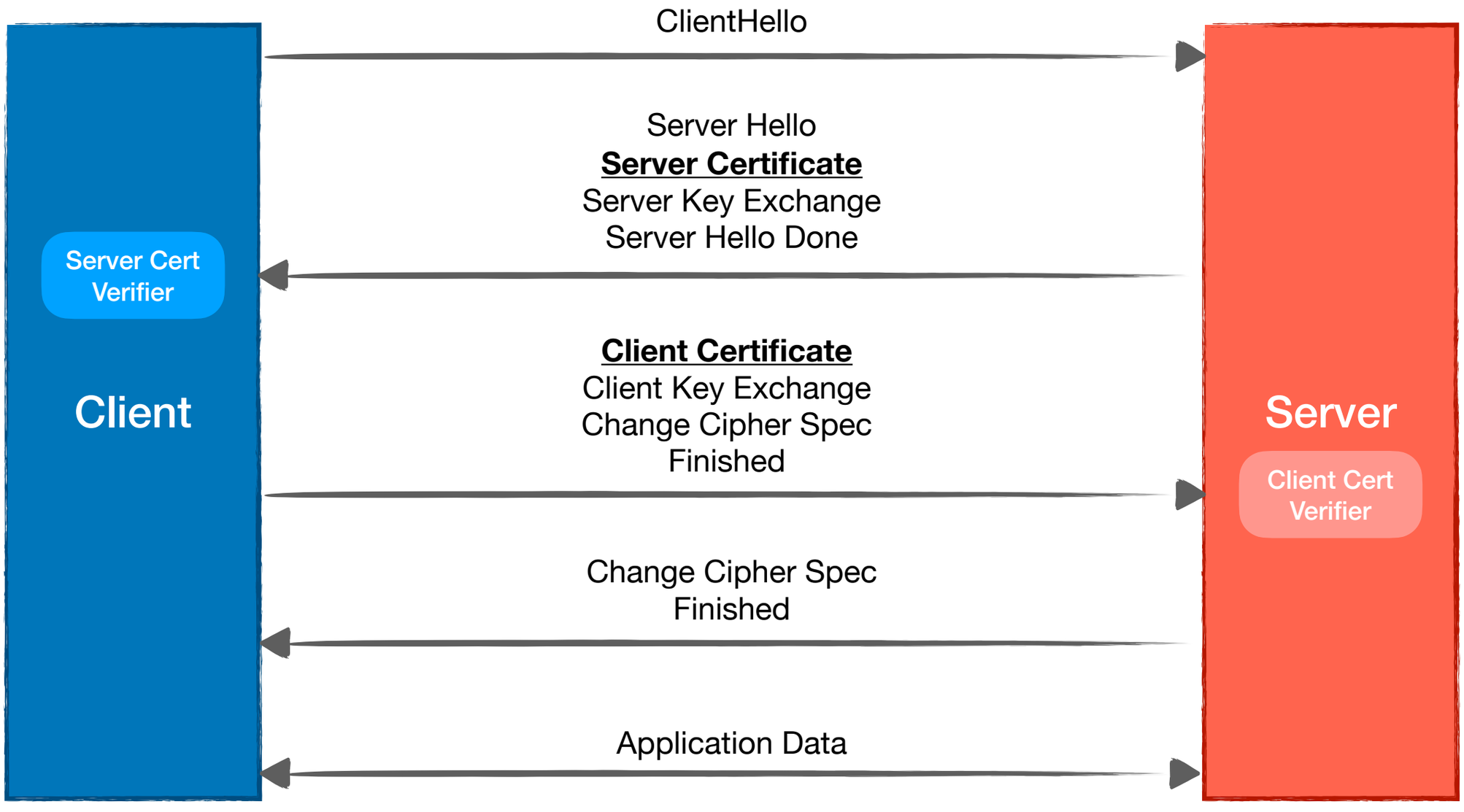
mics
- `/third_party`
- `/licenses`
- `/docs`
- `/keys`

api endpoint connections

internal endpoint connections

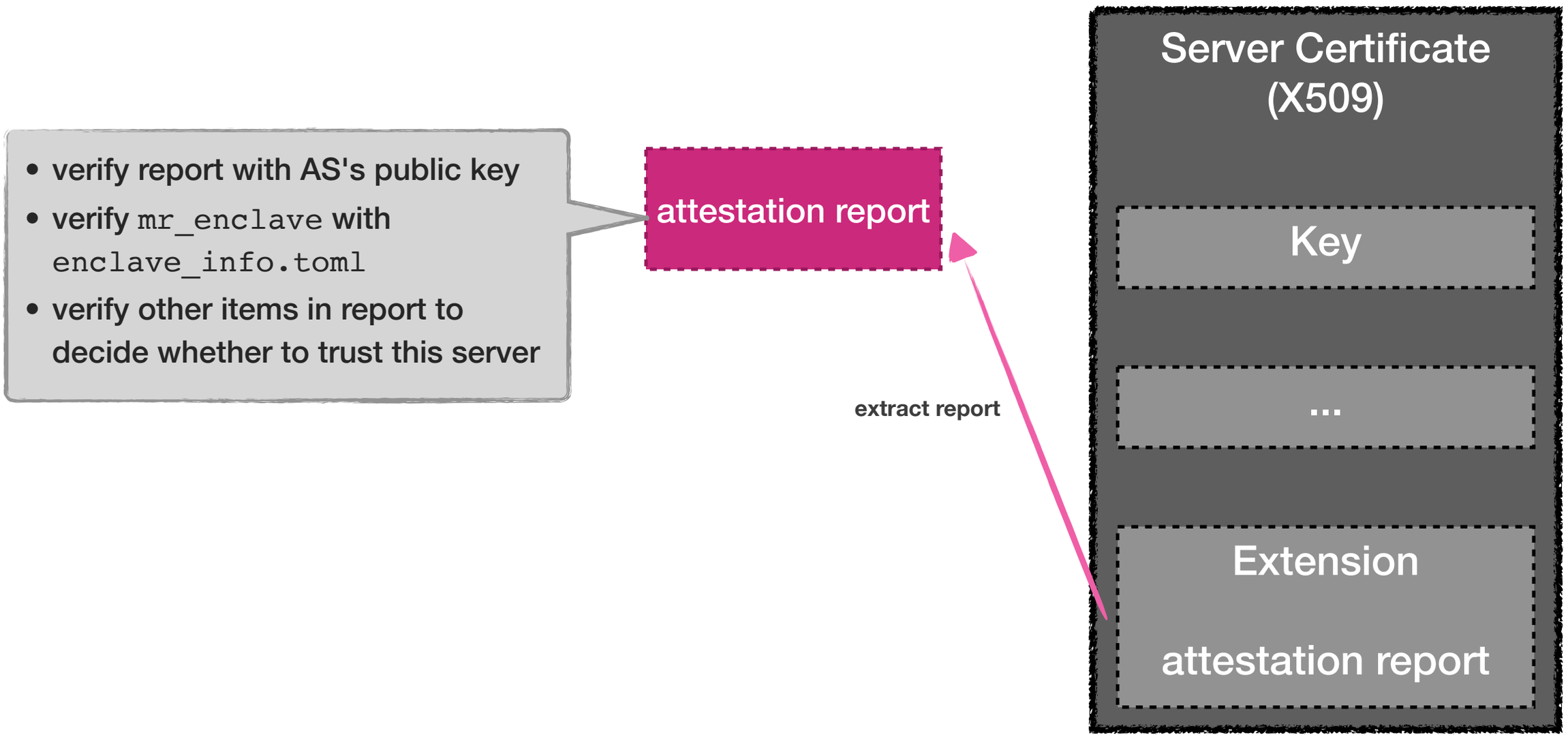
Attestation

- Attestation in TLS handshake



Attestation

- Server certificate verifier



Attestation

- Server certifica

```toml
[teaclave_access_control_service]
mr_enclave = "41860a711767332f80cb49a0c31ab0d597e79296597383ed7b47aa93f4eb942a"
mr_signer  = "83d719e77deaca1470f6baf62a4d774303c899db69020f9c70ee1dfc08c7ce9e"
[teaclave_authentication_service]
mr_enclave = "d5350aad6065e6bcfee0a5753af4e180ca95cd726053f3d9cff914a83ebbd40d"
mr_signer  = "83d719e77deaca1470f6baf62a4d774303c899db69020f9c70ee1dfc08c7ce9e"
[teaclave_execution_service]
mr_enclave = "6ff6f7a2bd452b1602042b13876ecfdb6f7f8b5285fb5d080c656e14d49b08fc"
mr_signer  = "83d719e77deaca1470f6baf62a4d774303c899db69020f9c70ee1dfc08c7ce9e"
```

- 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 report

```
→  auditors git:(develop) tree
.
├── albus_dumbledore
│   ├── albus_dumbledore.public.pem
│   └── albus_dumbledore.sign.sha256
├── godzilla
│   ├── godzilla.public.pem
│   └── godzilla.sign.sha256
└── optimus_prime
    ├── optimus_prime.public.pem
    └── optimus_prime.sign.sha256
```

extract repo

attestation report

- `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.

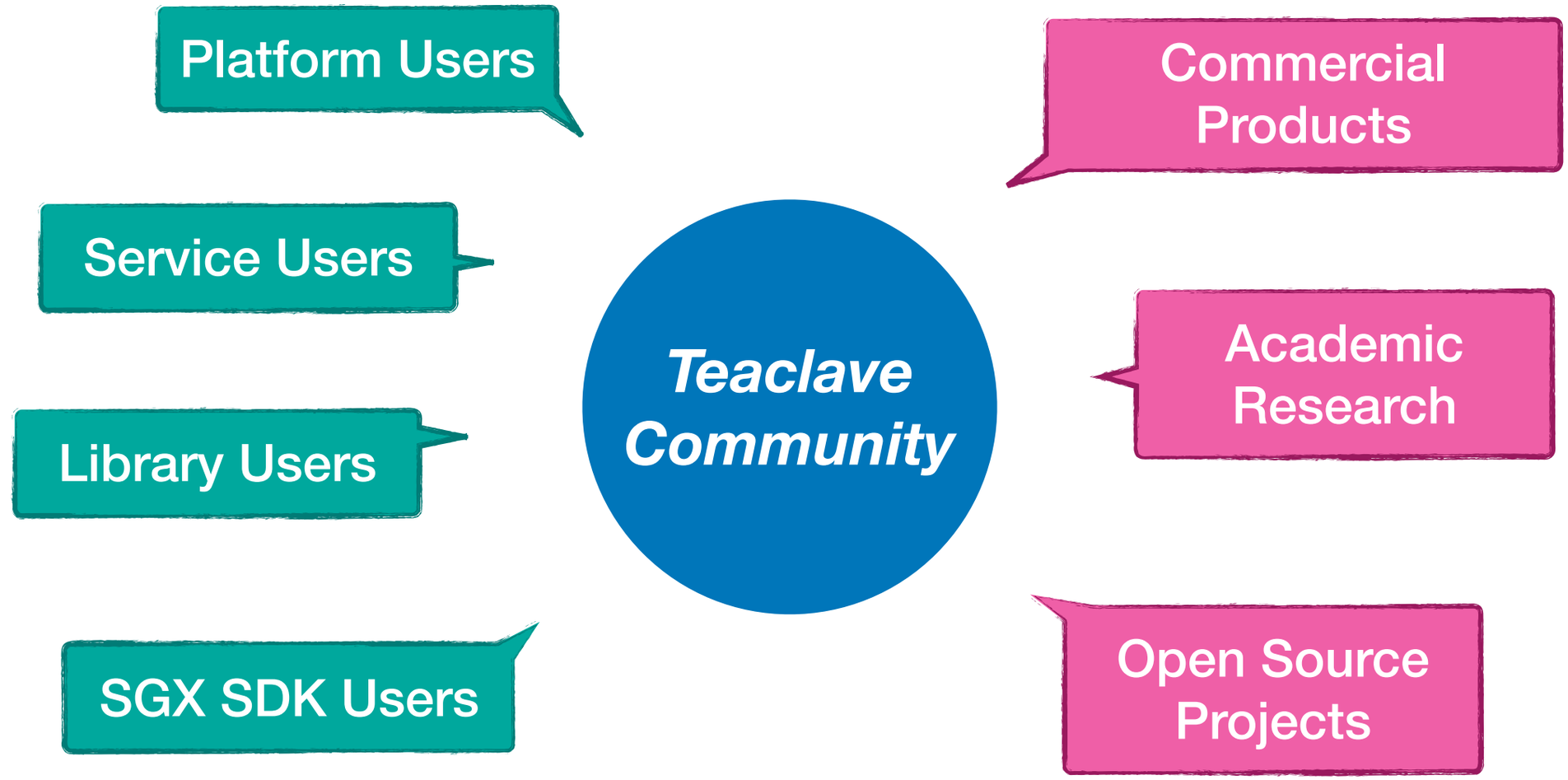
Getting Started

- **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
  - Teaclave Service Internals
- **Contribute**
  - Rust Development Guideline
  - Development Tips
- **Codebase**

Documentation

https://teaclave.apache.org/docs/

Teaclave Community



Teaclave Community

Organizations

- Baidu
- ByteDance
- Enigma
- LayerX

https://teaclave.apache.org/community/

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.

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!**