Confidential Computing

with SCONE - part 1

Christof Fetzer
https://sconedocs.github.io

Confidential Computing

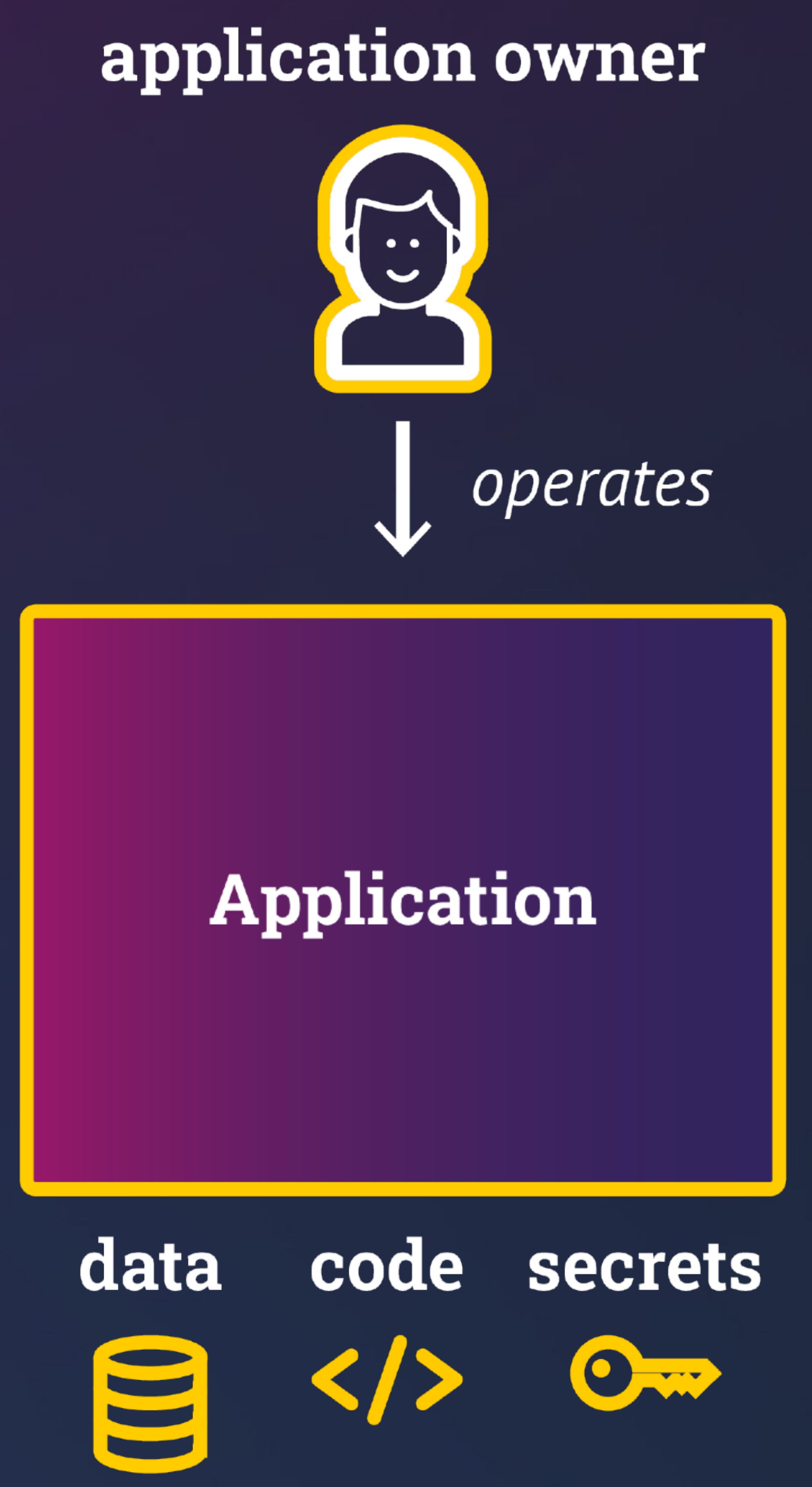
- Motivation -

Motivation

- **Role**: **application owner**

Objectives:

- provides an application to clients
- protects **data**, **code**, and **secrets** of the application



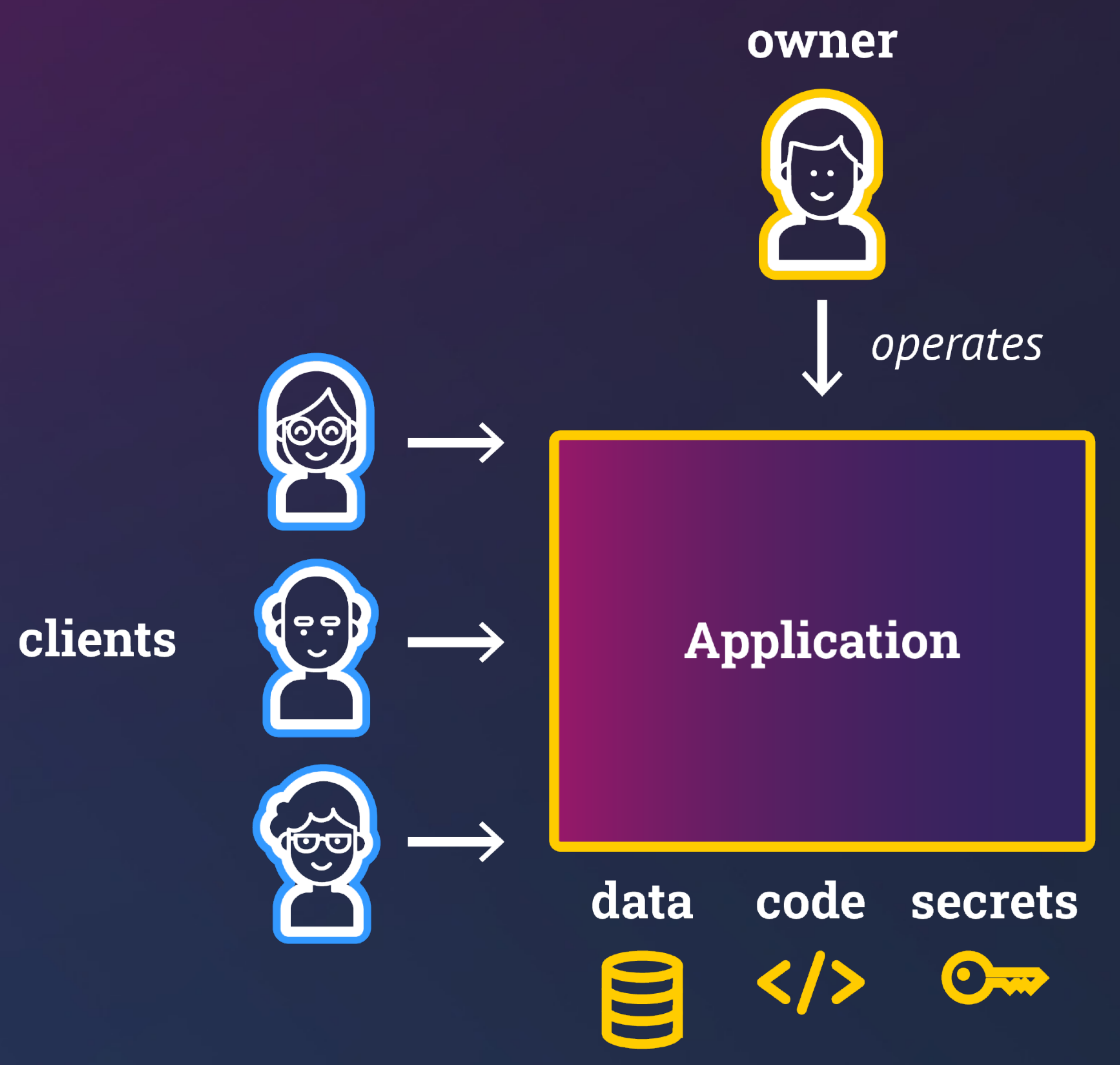
Motivation

- **Role: application owner**

Objectives:

  - provides an application to clients
  - protects **data, code, and secrets** of the application

- **Role:** clients
  - can connect to the application
  - access their data



Requirements

- example: eHealth domain -

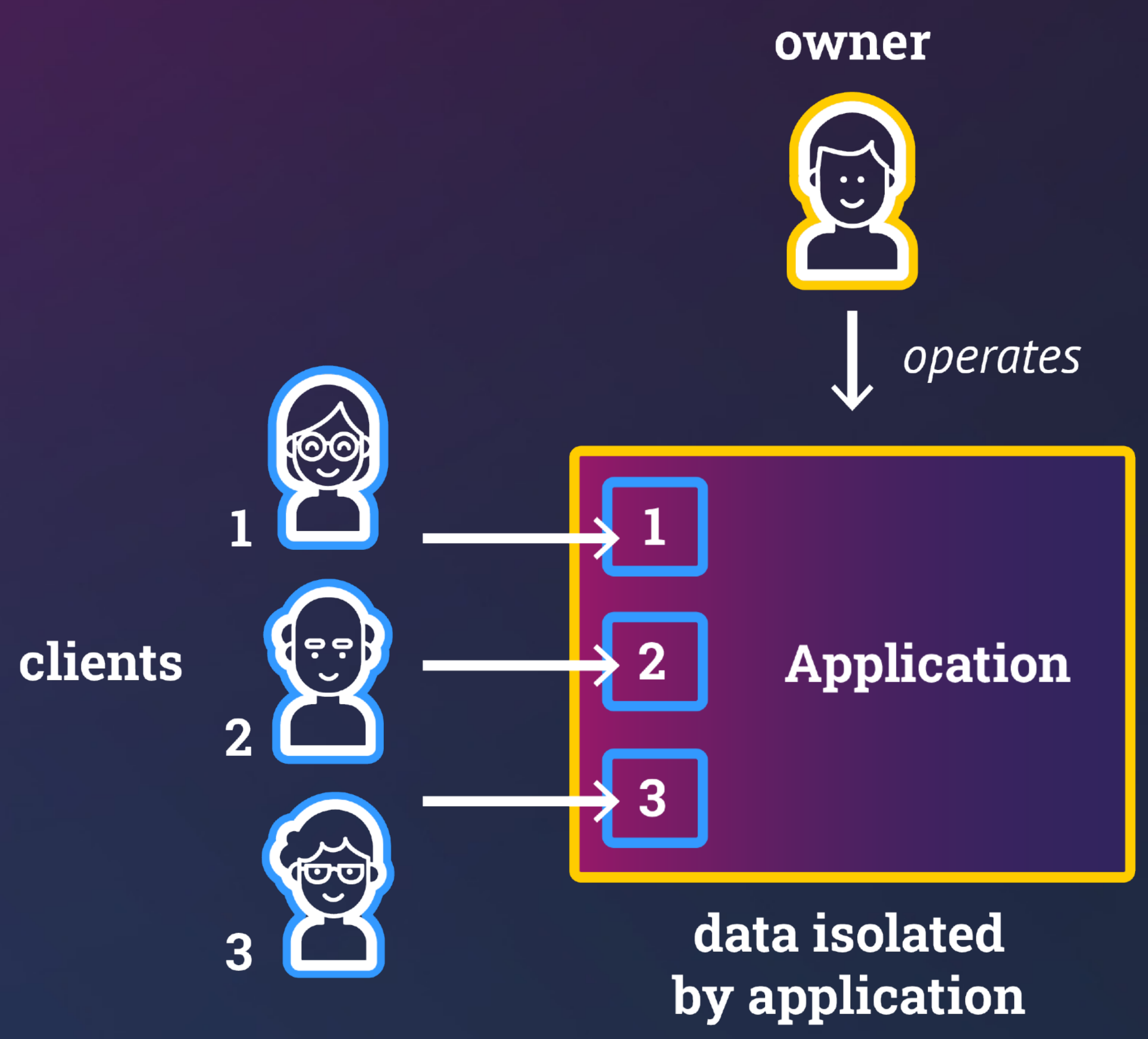
Example Requirement: Isolation of data

- **Role: application owner**

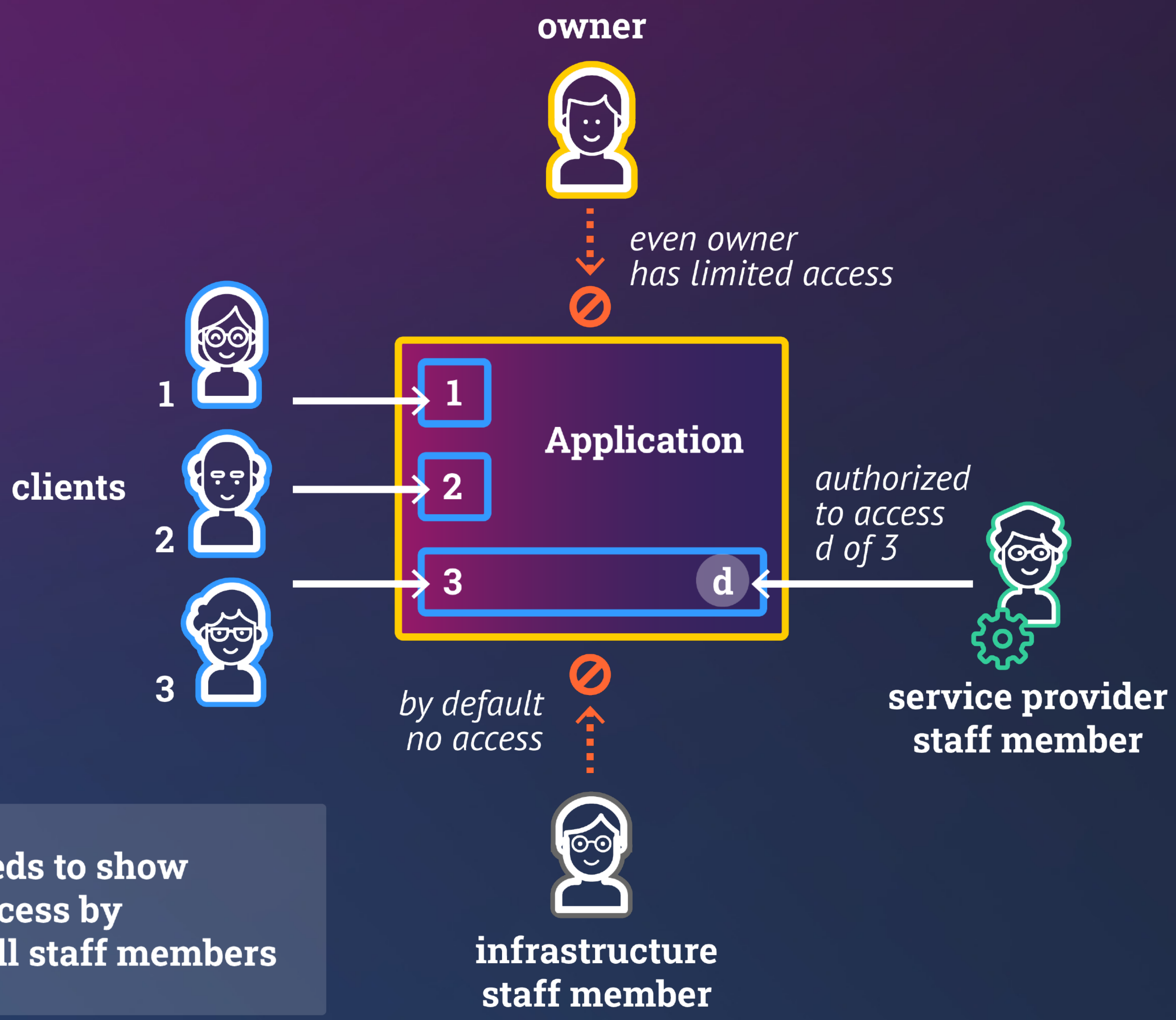
Objectives:

- provide an application to clients
- protect **data, code, and secrets** of the application

- **Role:** clients
  - can connect to the application
  - access their data
  - **application isolates data of clients**

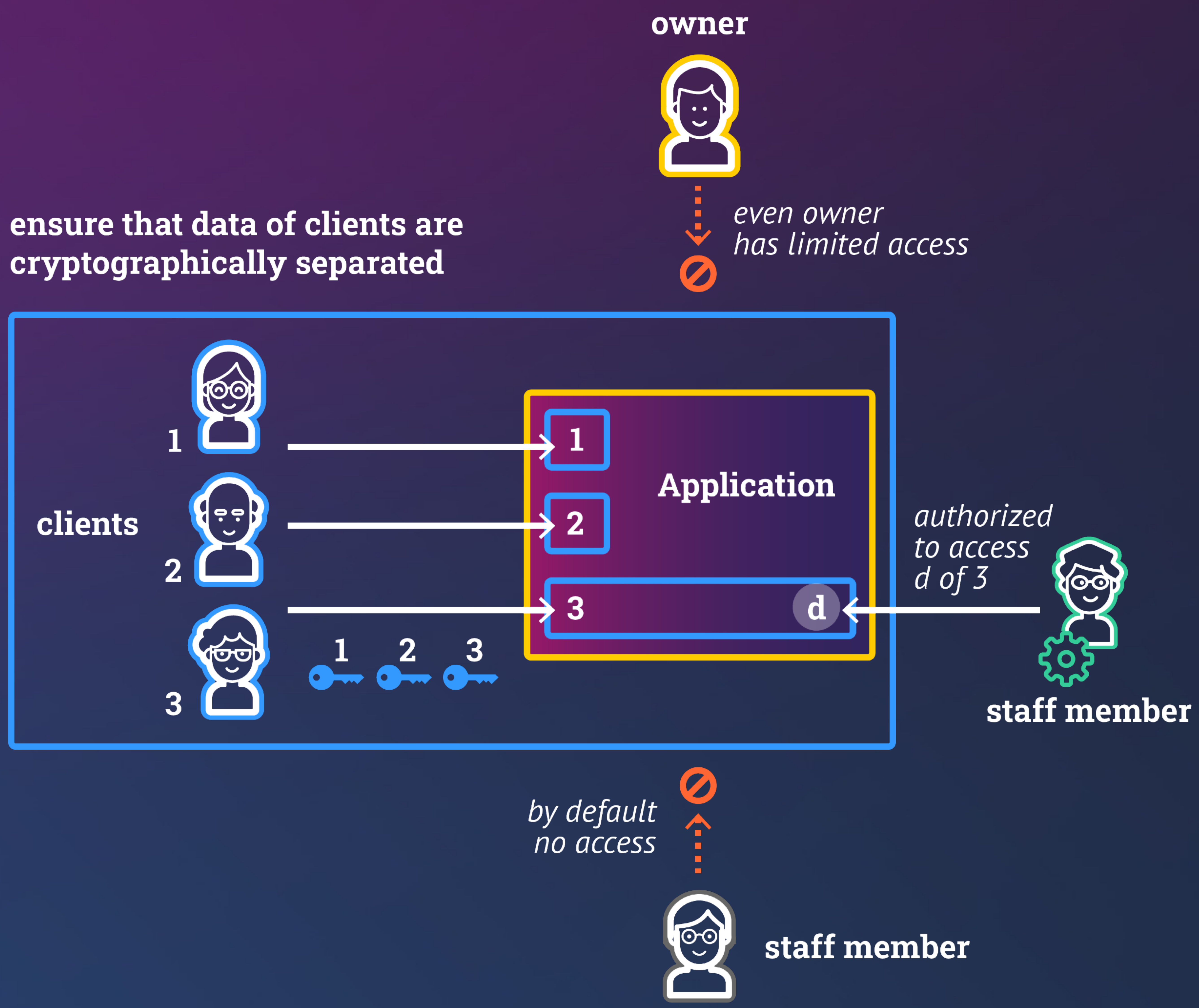


Limited Access by Owner & Staff

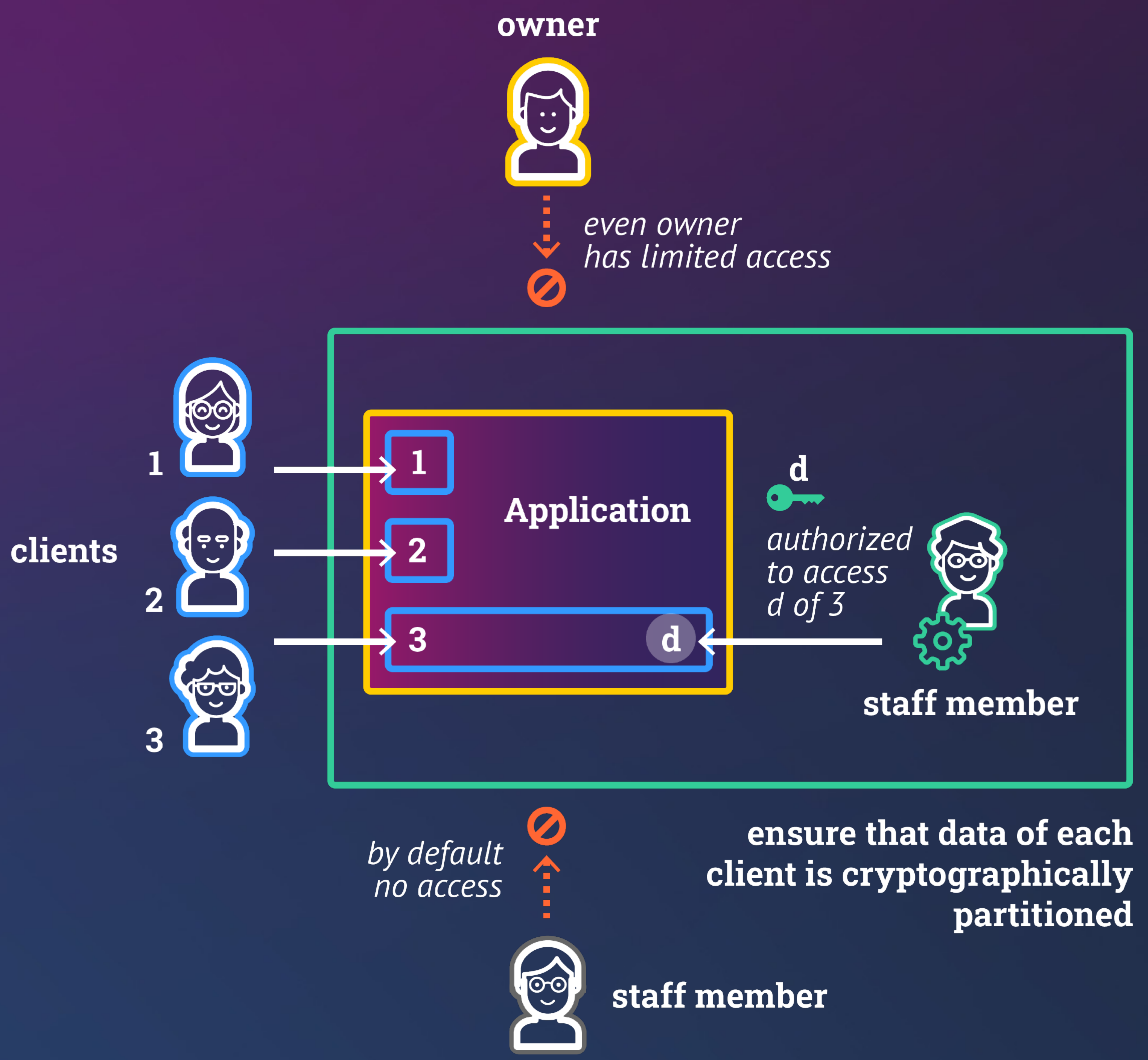


owner needs to show
limited access by
owner & all staff members

Divide and Conquer



Divide and Conquer



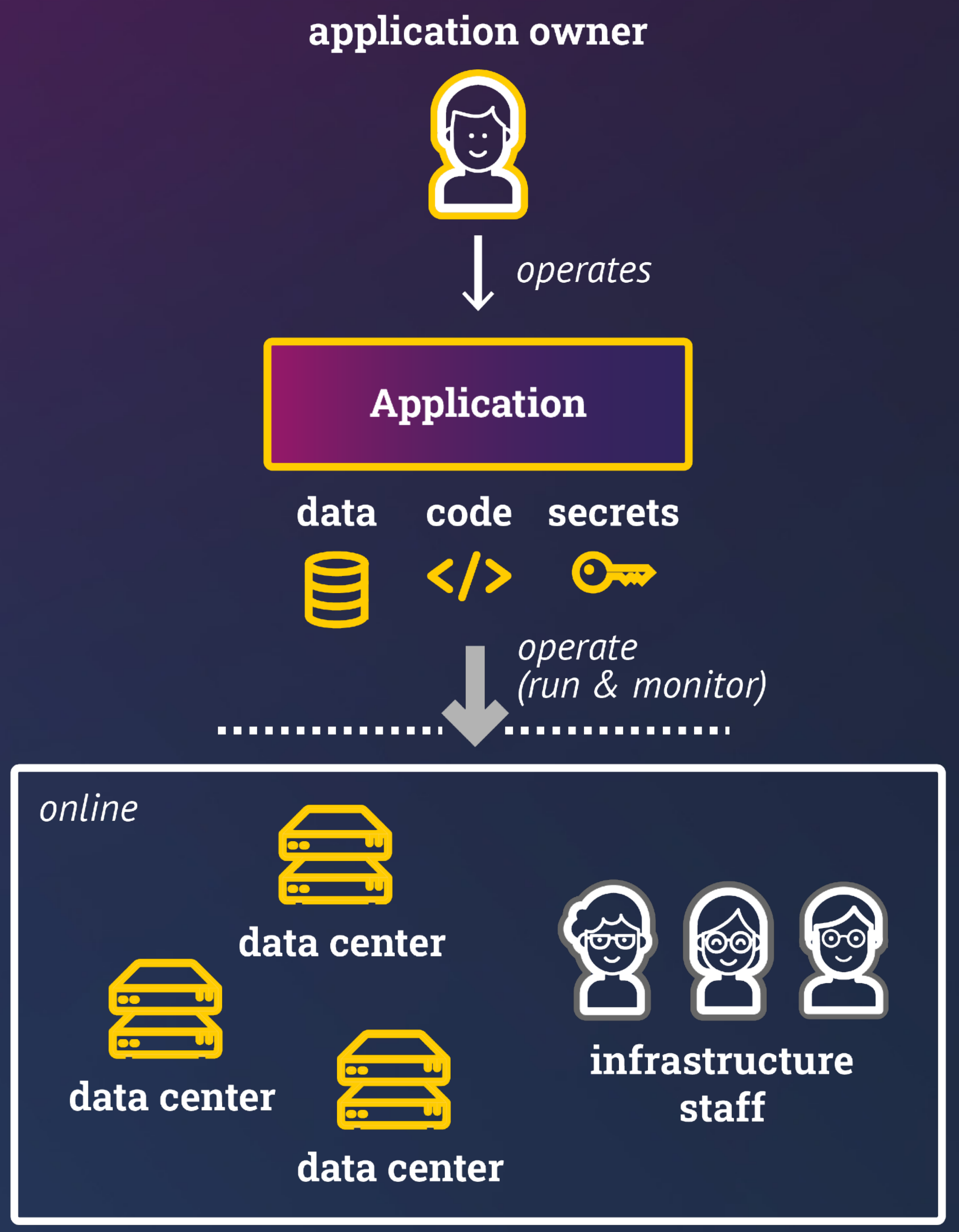
ensure that data of each client is cryptographically partitioned

Business Problem

Problem Description

Problem: application owner cannot operate the application

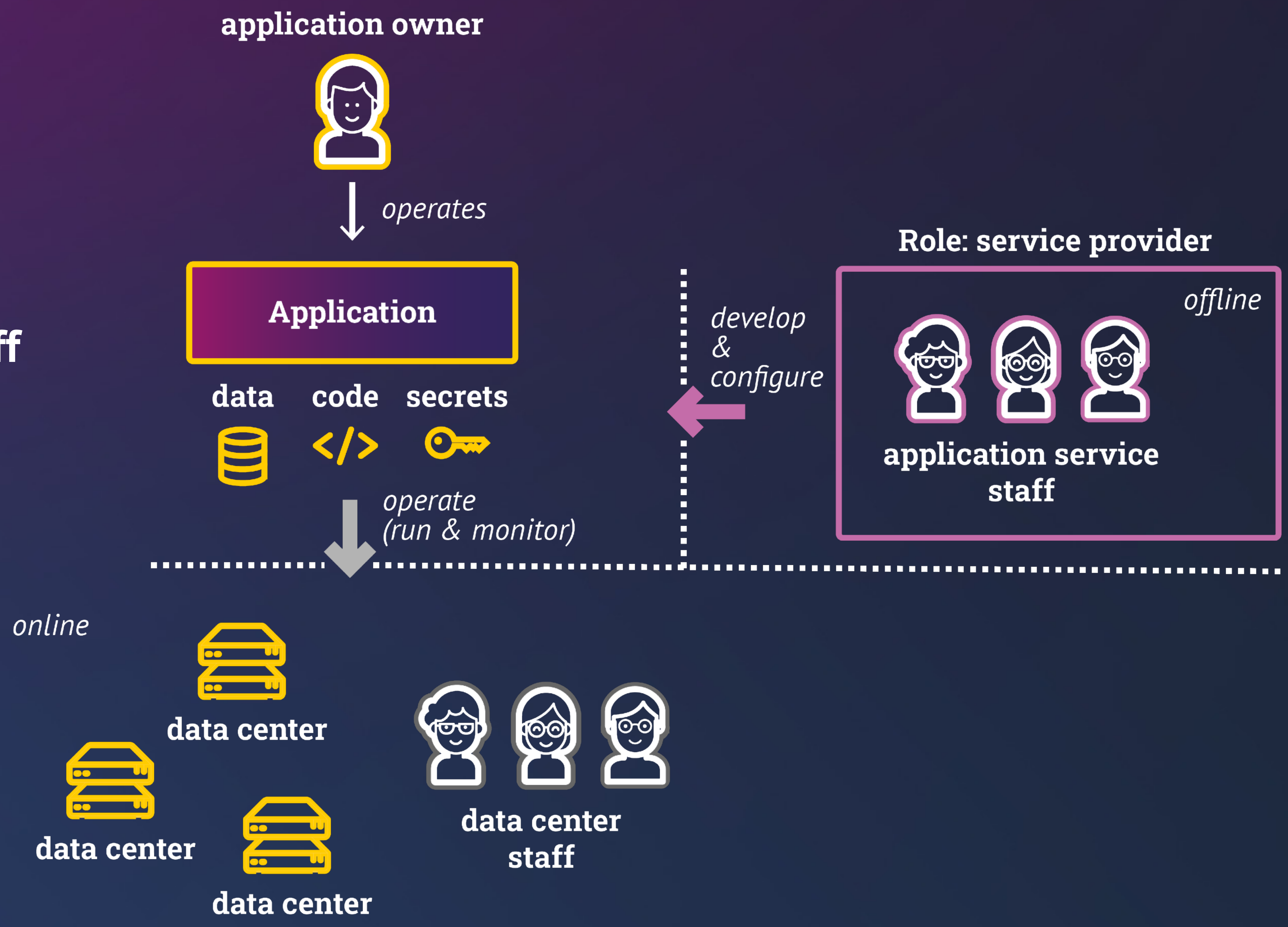
- **lack of data centers || trusted infrastructure staff**
- lack of application service staff



Problem Description

Problem: application owner cannot operate the application

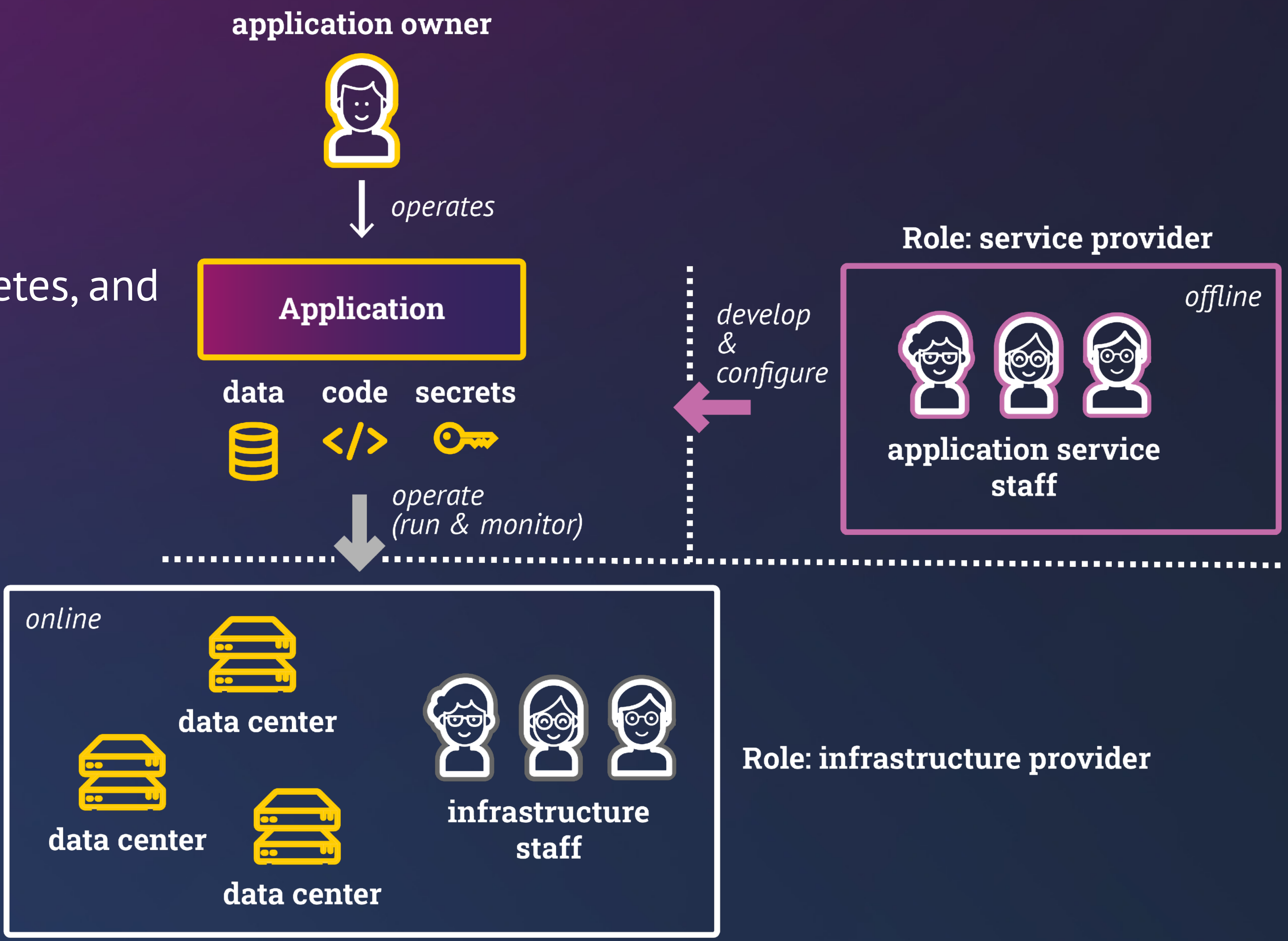
- lack of data centers || trusted infrastructure staff
- **lack of application service staff**



Approach: Outsource!

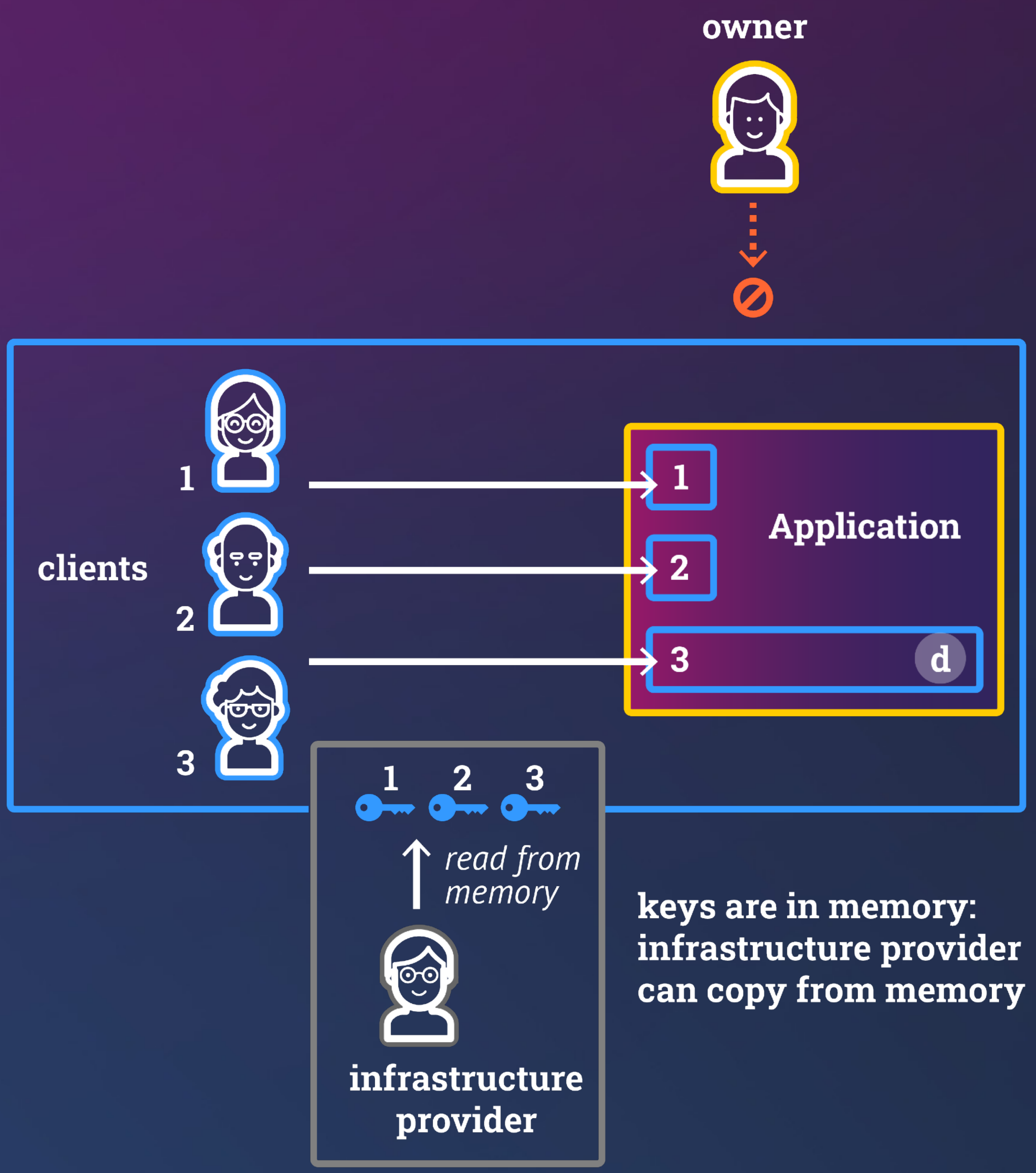
Approach: external entities

- operate data centers, Kubernetes, and
- manage application development



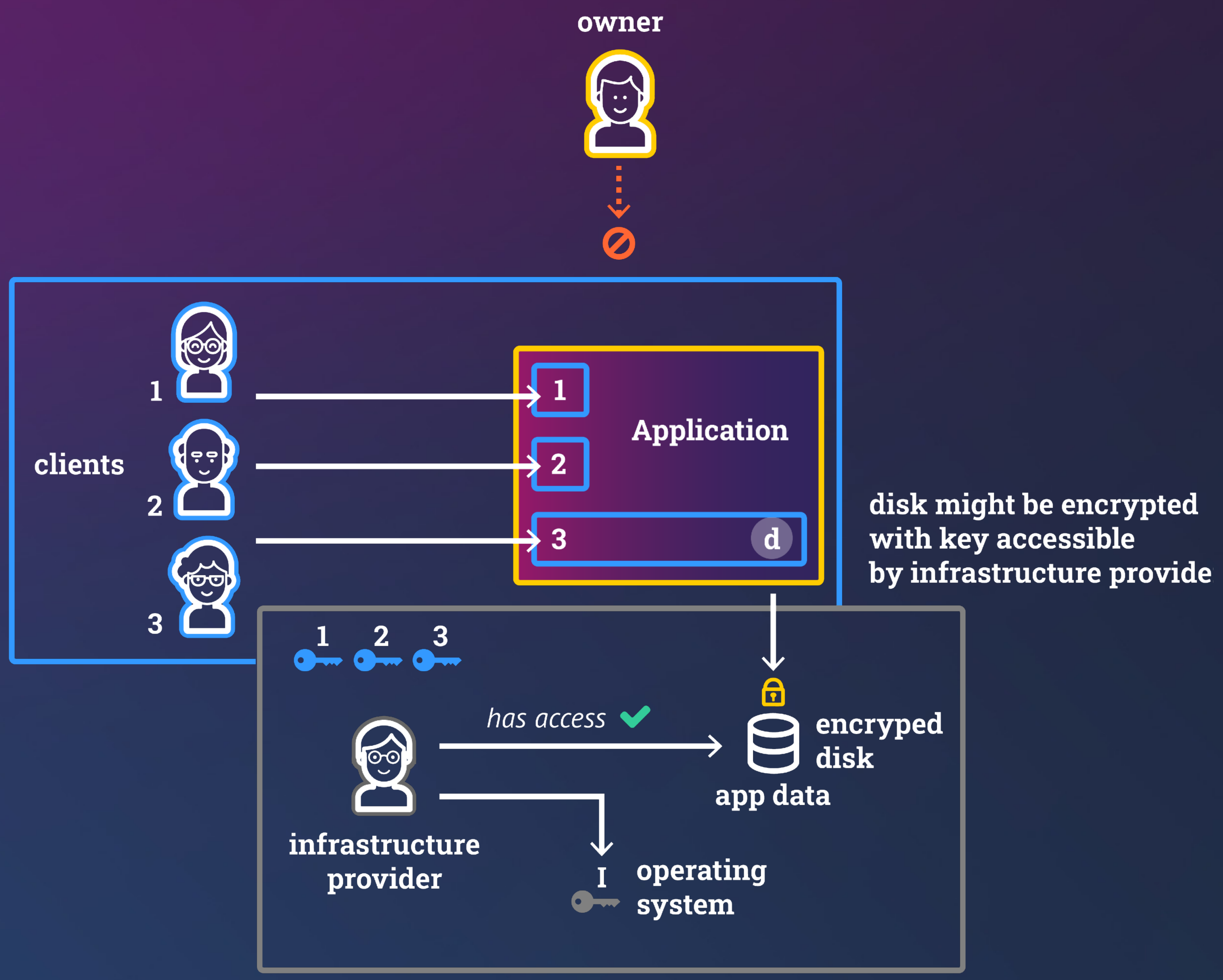
Technical Problem Description

Problem: Hardware & Admin Access

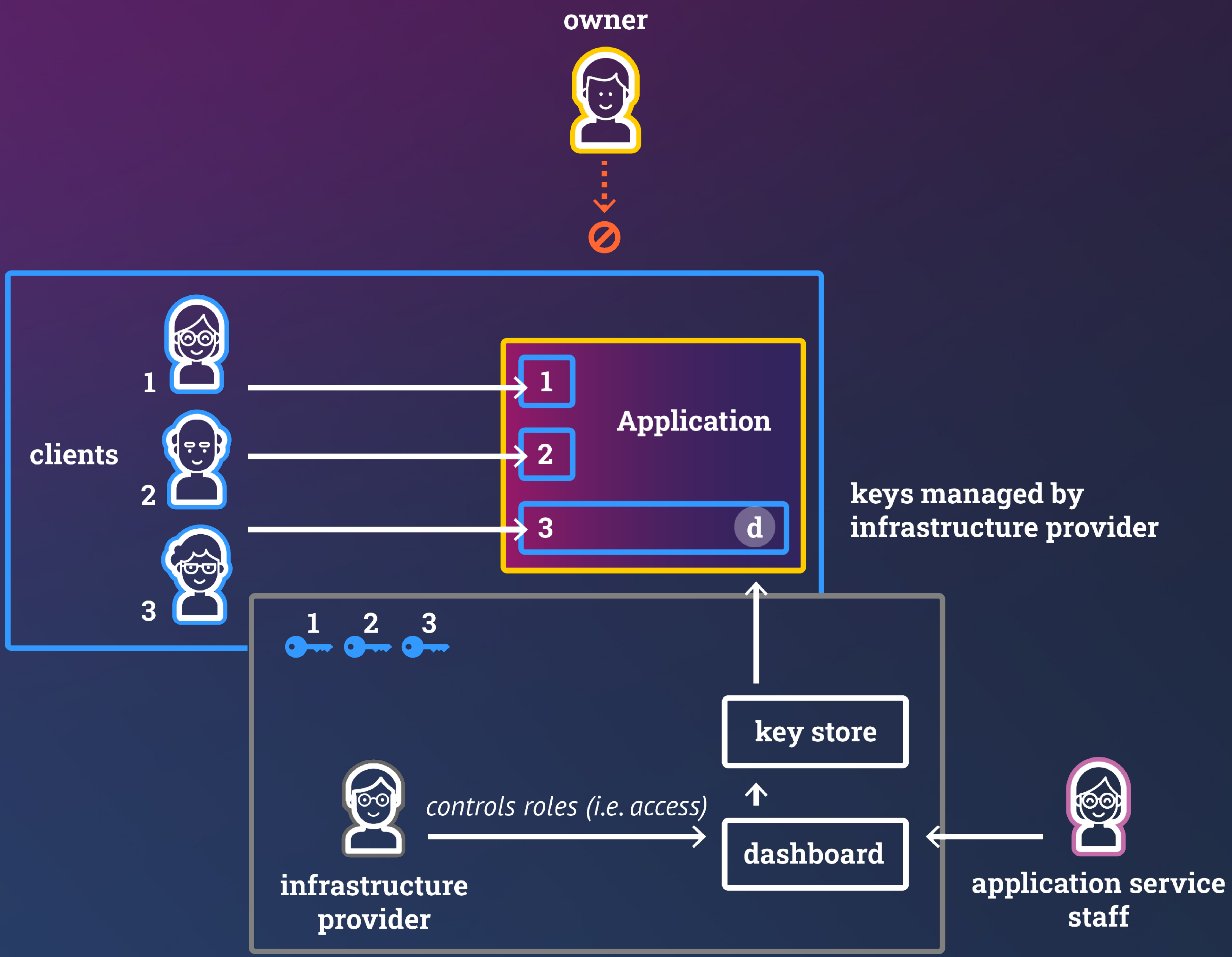


keys are in memory:
infrastructure provider
can copy from memory

Problem: Encrypted Disks



Problem: Key Management



Threat Model & Implications

- we might trust the CPU -

Who to trust?

owner

infrastructure quickly evolving, application owner cannot vouch for security

Implication:
We need to ensure no access to source code, data or any keys

infrastructure provider

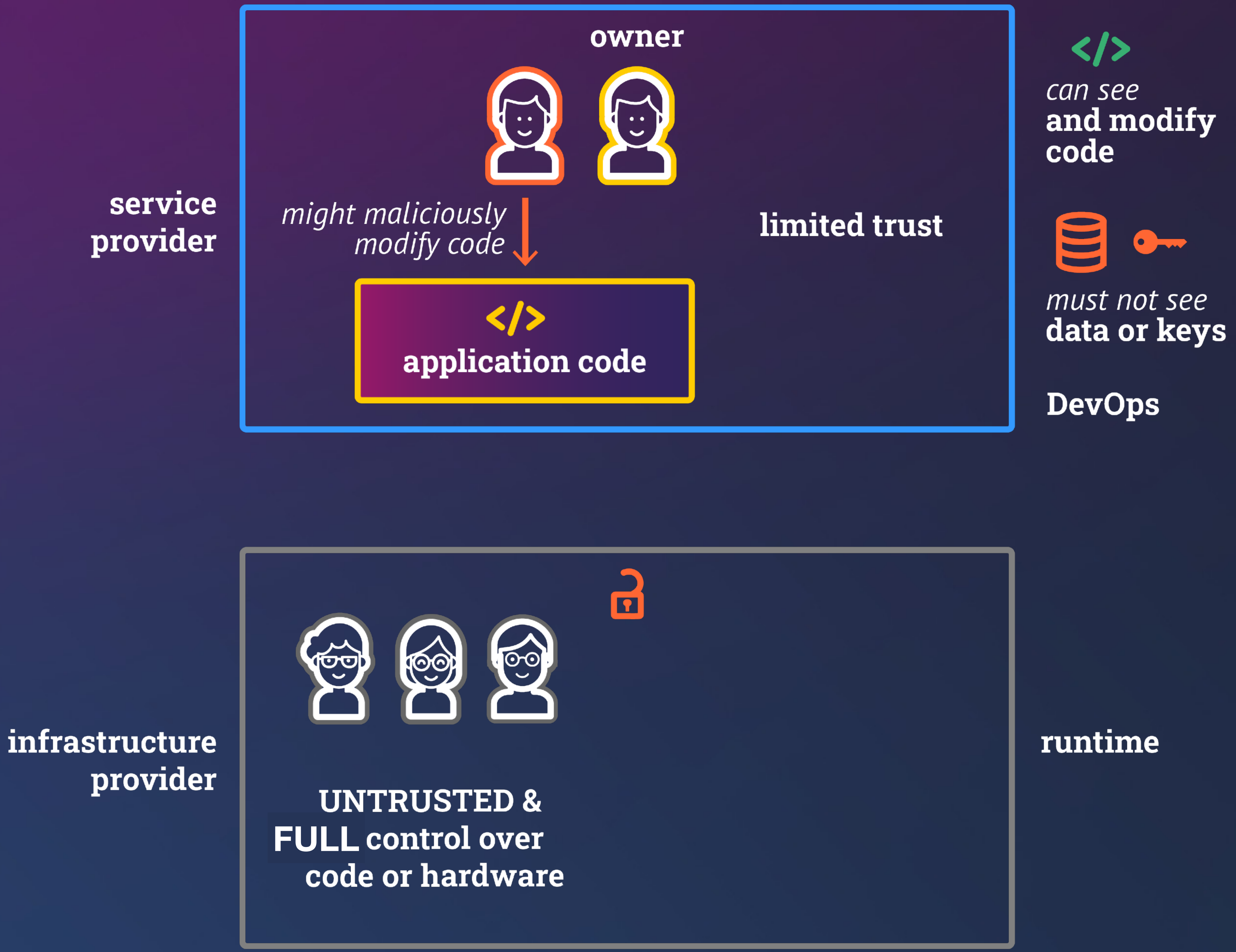
UNTRUSTED &
FULL control over
code or hardware

managed hypervisor,
operating system,
Kubernetes,
key store,
access control,
...staff members

are ALL UNTRUSTED

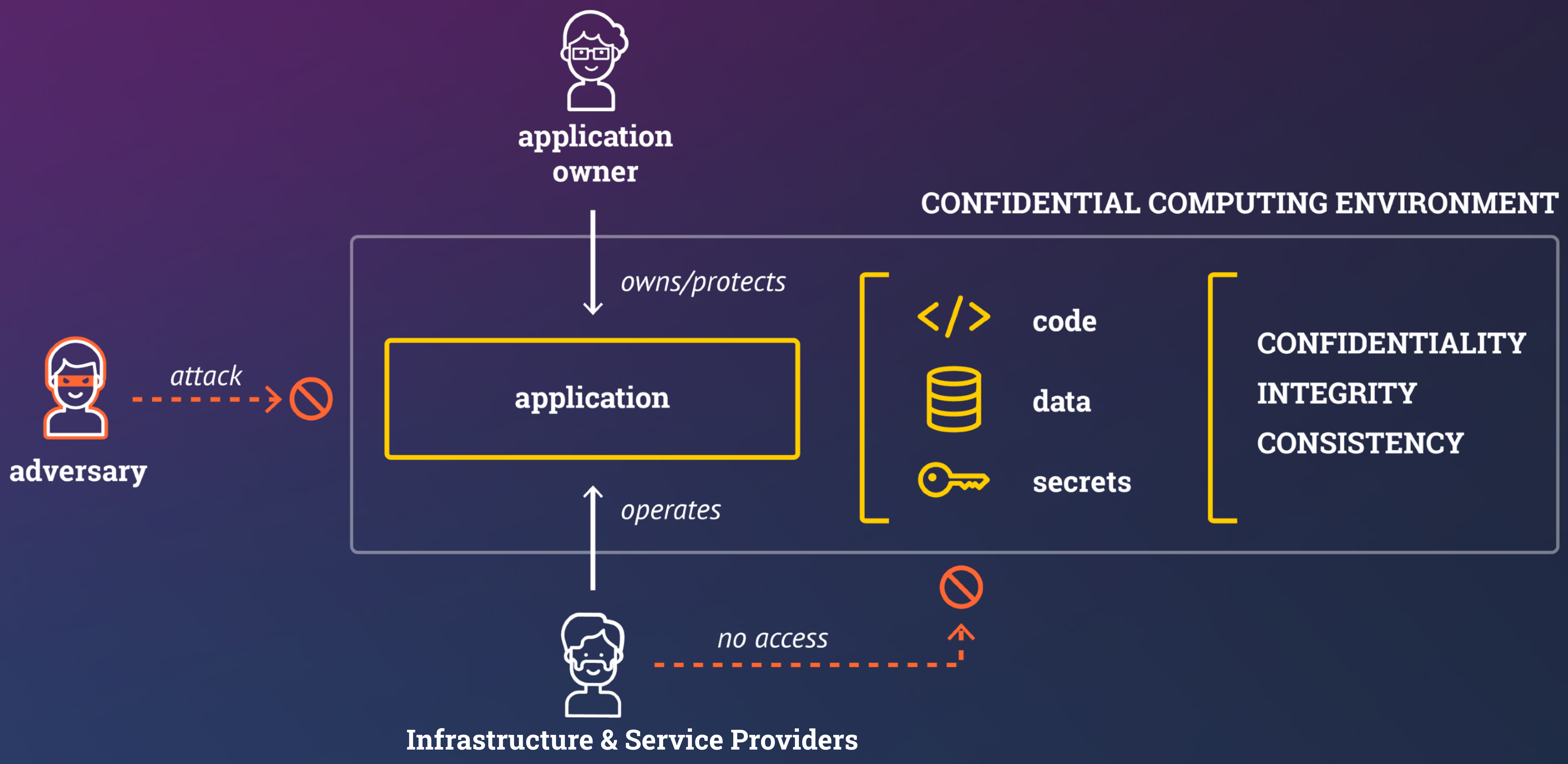
runtime

Threat Model



Approach

Approach: Confidential Computing



„application-oriented security“

Confidential Computing

with SCONE
Some basic concepts and terminology

Christof Fetzer
https://sconedocs.github.io

Application Domains of Confidential Computing

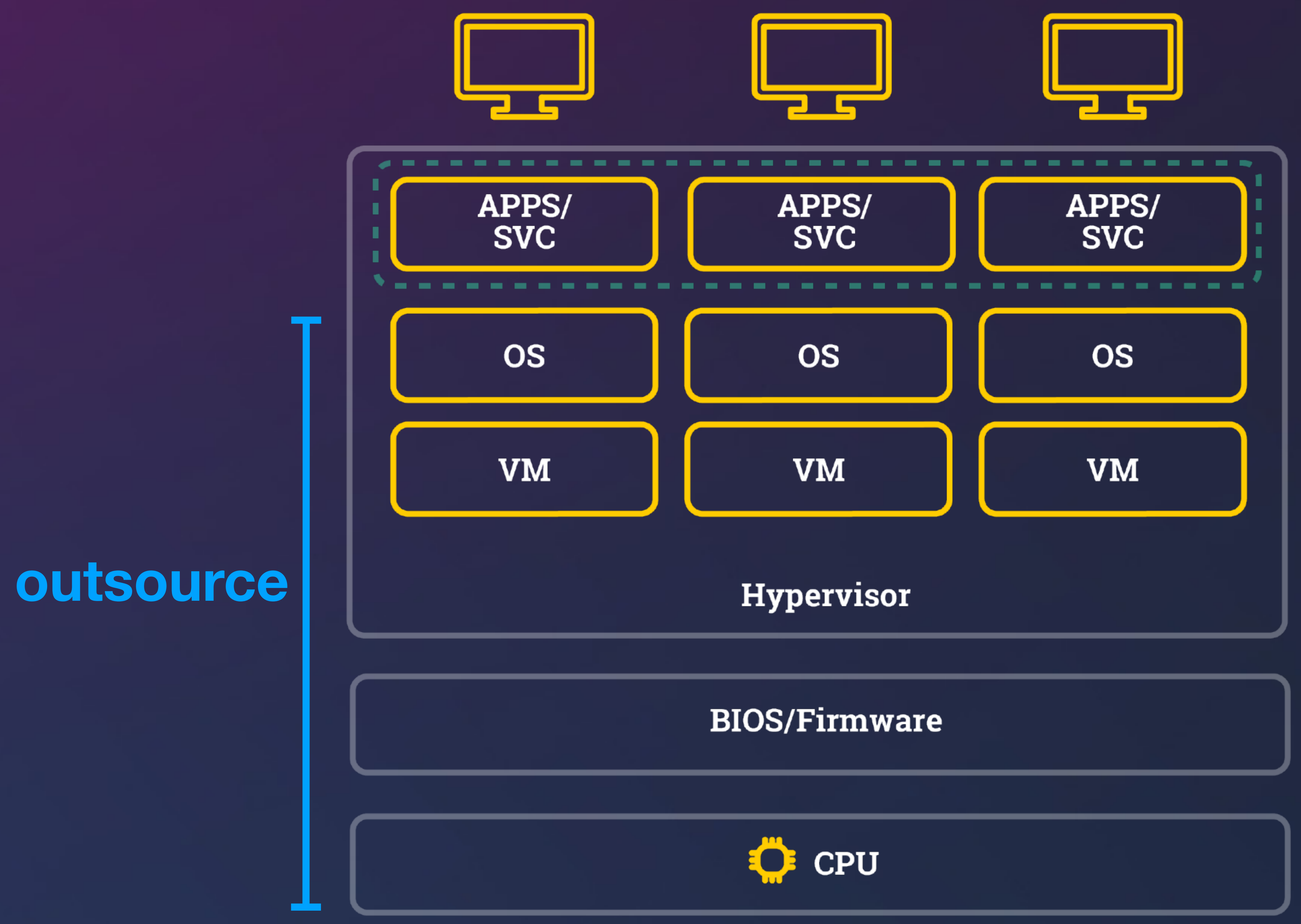
Confidential Computing: Application Domains

- **Cloud computing**
  - Protect cloud-native applications
  - Protect AI applications
  - Protect cloud services itself
- **Edge cloud**
  - Protect applications and cloud
    - despite limited physical security
- **Air-Gapped systems**
  - Protect applications despite having no connectivity
- **Embedded systems**
  - Protect devices with limited connectivity & limited physical security
- **eHealth domain**
  - e.g., protect patient data

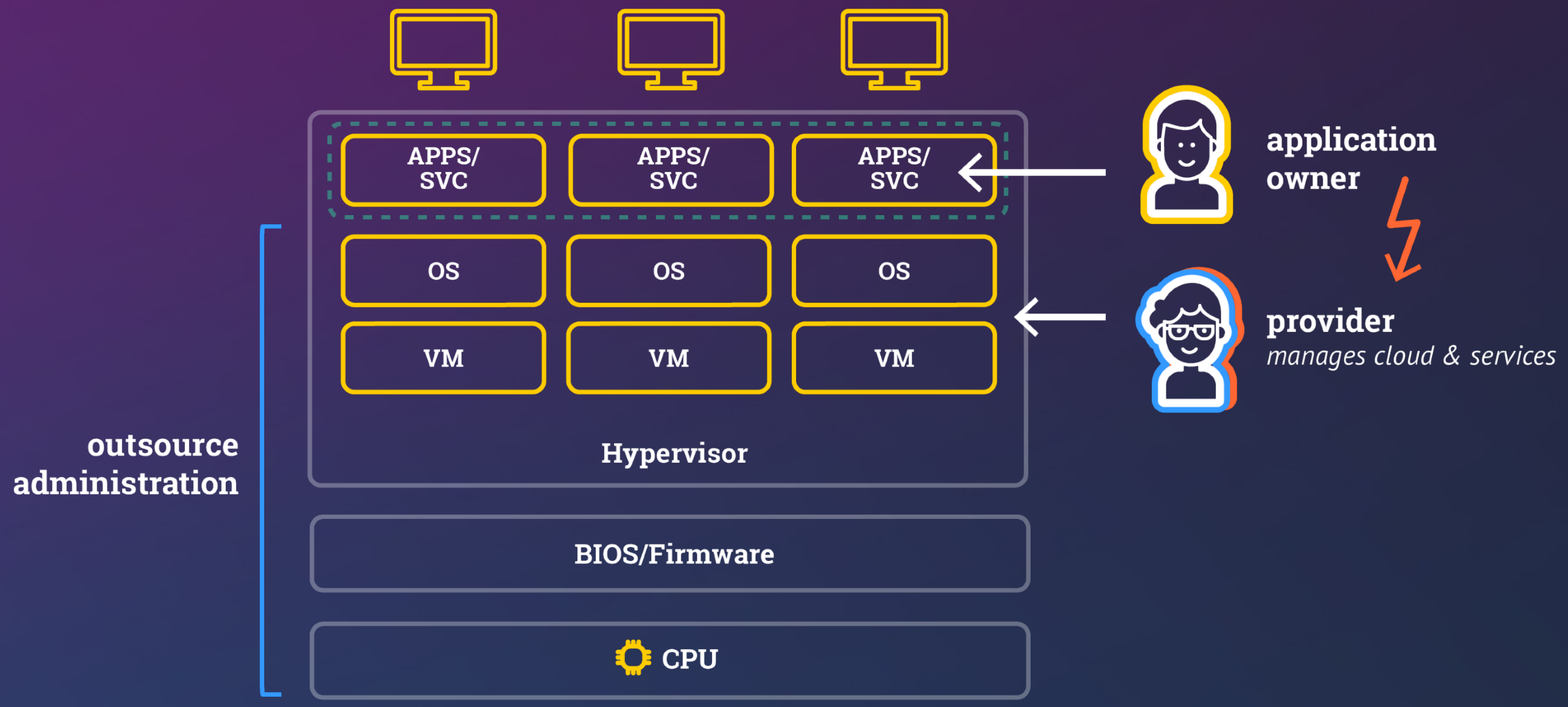
Cloud: Outsource Infrastructure

Operating one's own computing infrastructure is not easy:

- data center
- hypervisors
- operating systems
- Kubernetes
- services
- ...



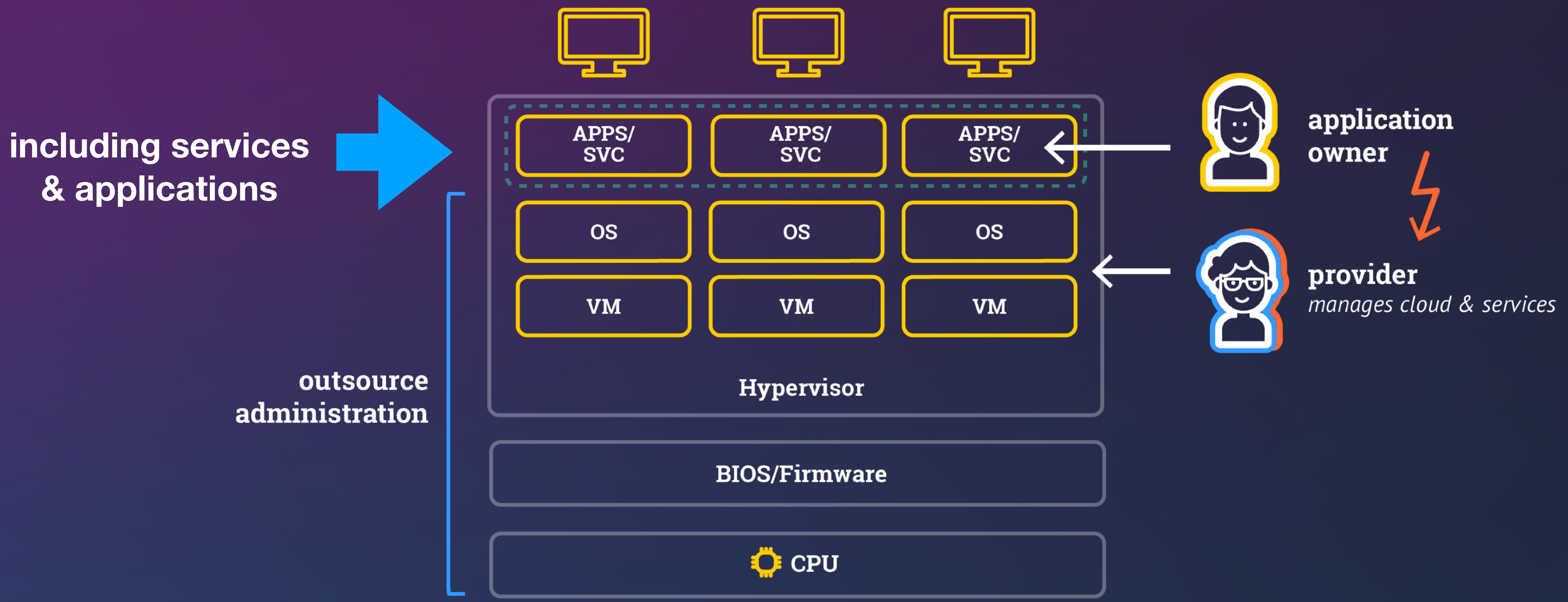
Different Stakeholders



protects application

protects system

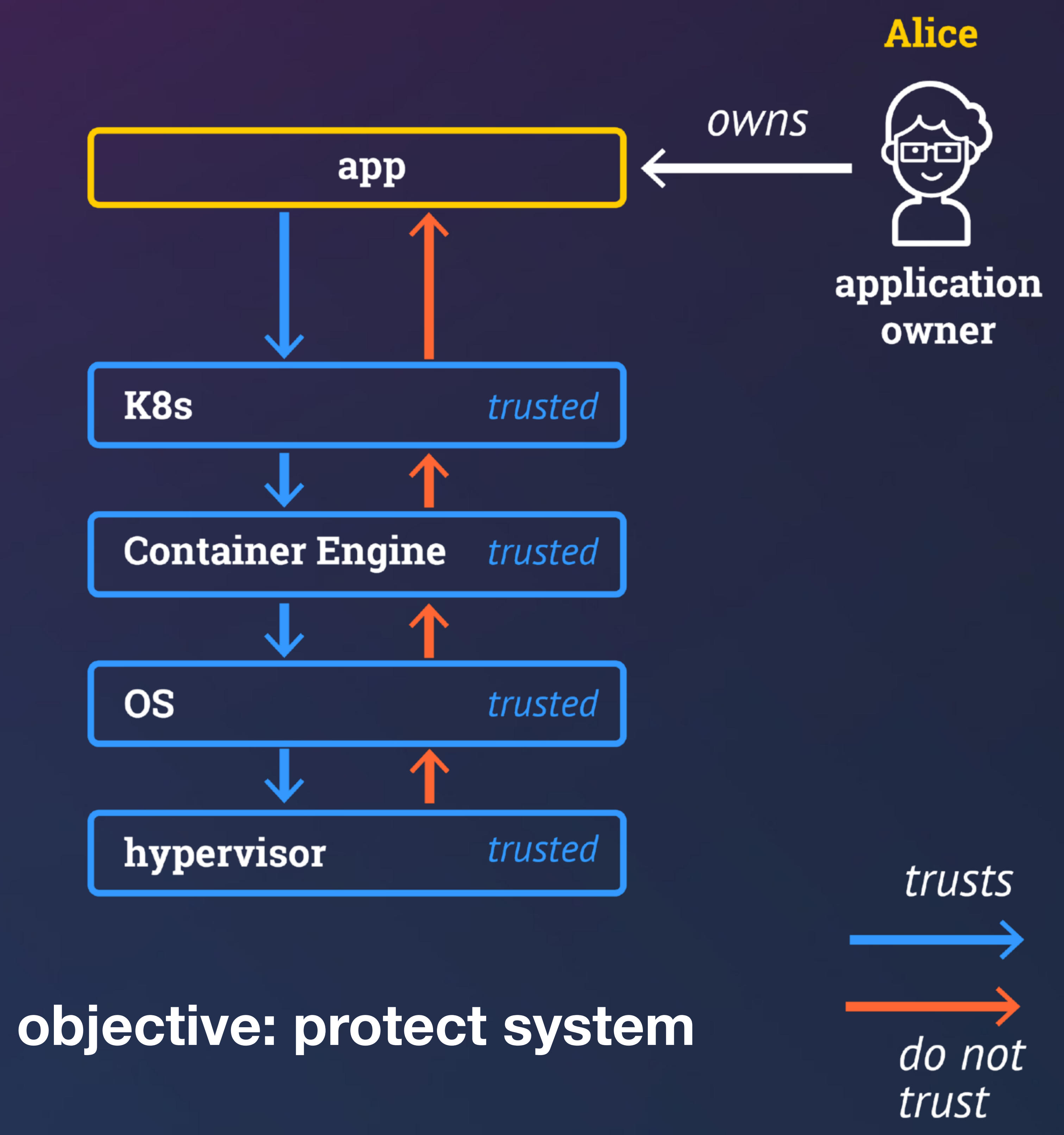
Confidential Managed Services



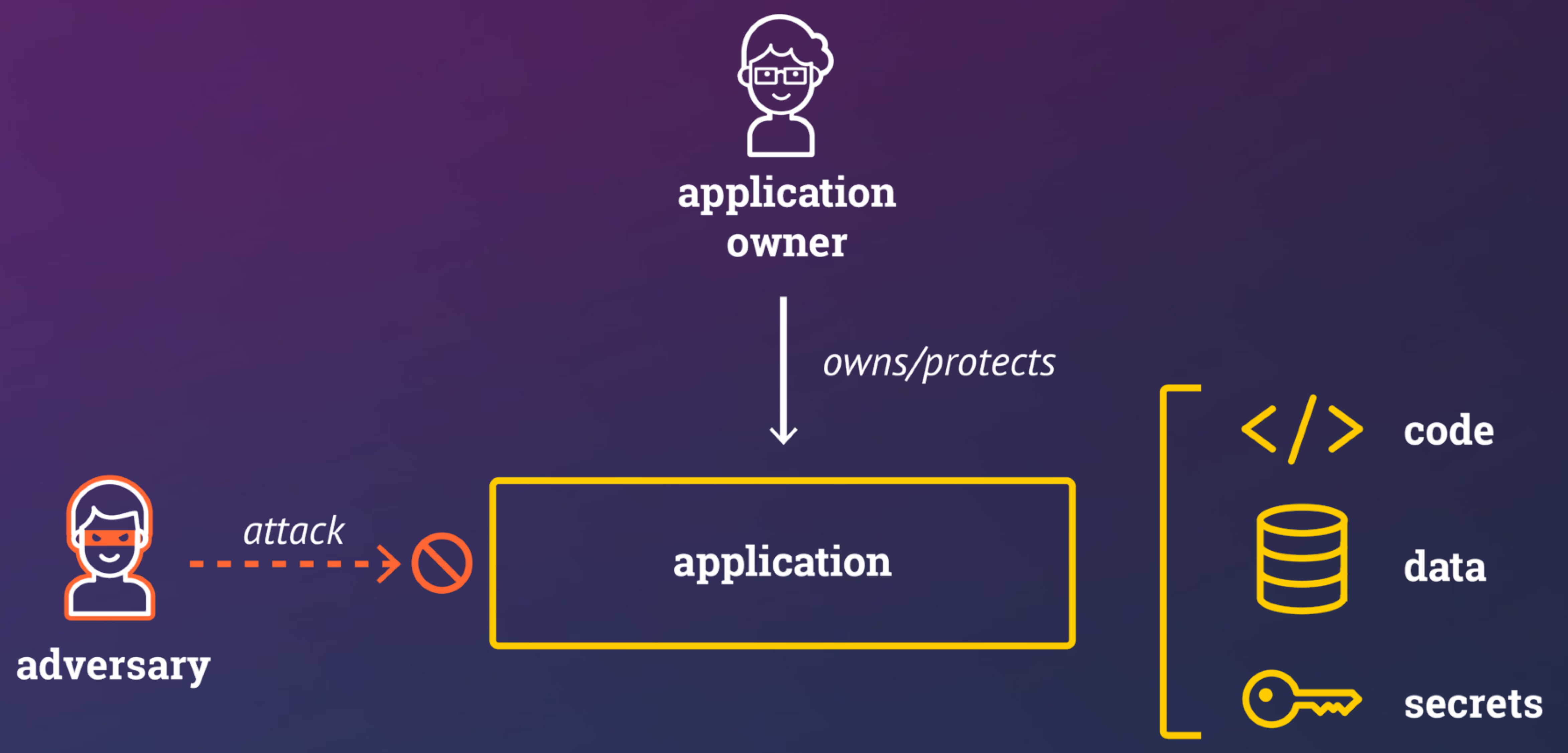
Objective: Outsource management of services and applications provider must not have access to data / code / secrets.

Systems Security

- **Systems are structured in layers**
    - like operating system and hypervisor
- **Typically, systems security is bottom up**
    - layer i does not trust layer i+1
    - but layer i+1 trusts layer i
- **Examples:**
    - the operating system trusts the hypervisor
    - but the hypervisor does not trust the operating system

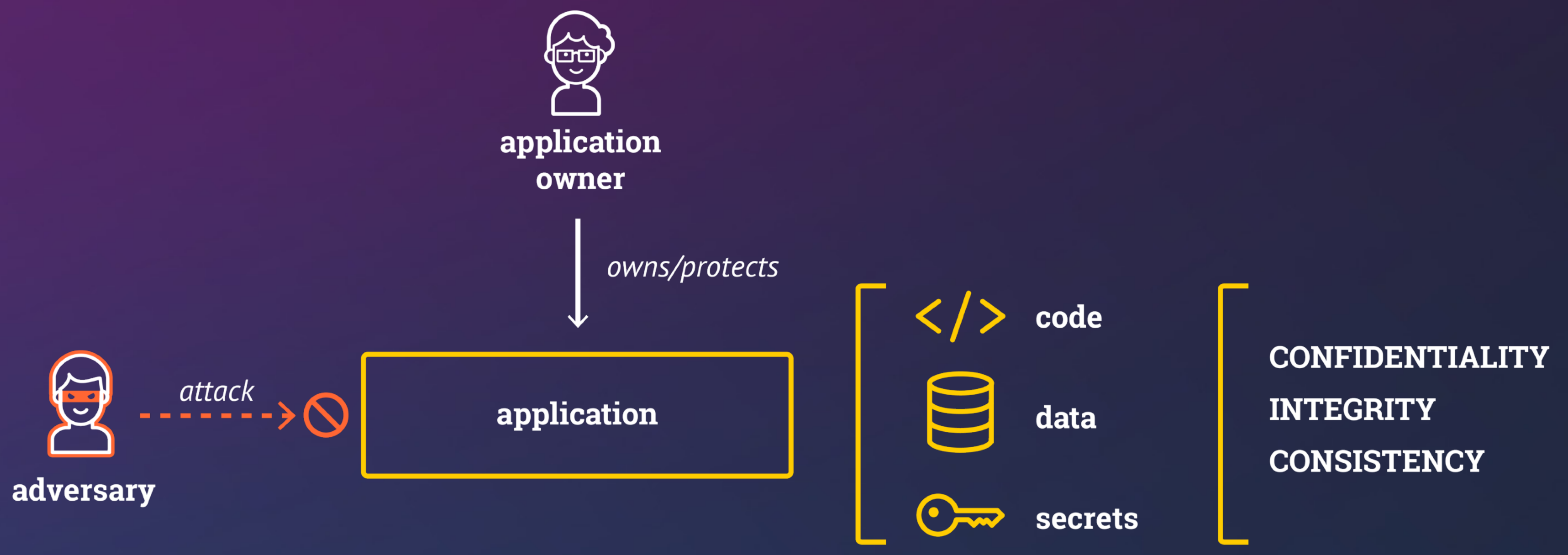


Confidential Computing: Protect Applications



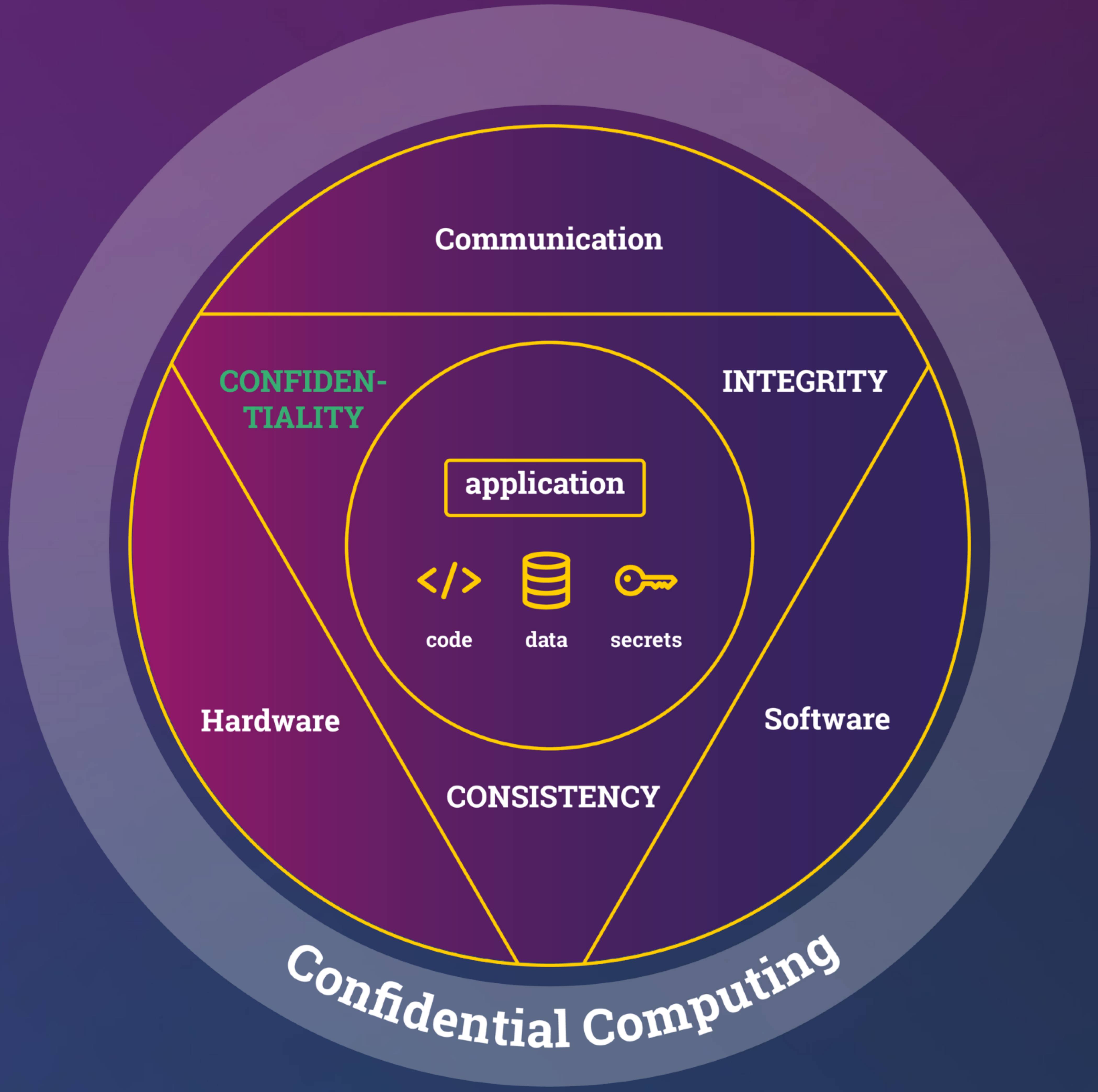
„application-oriented security"

Protect Code, Data, and Secrets



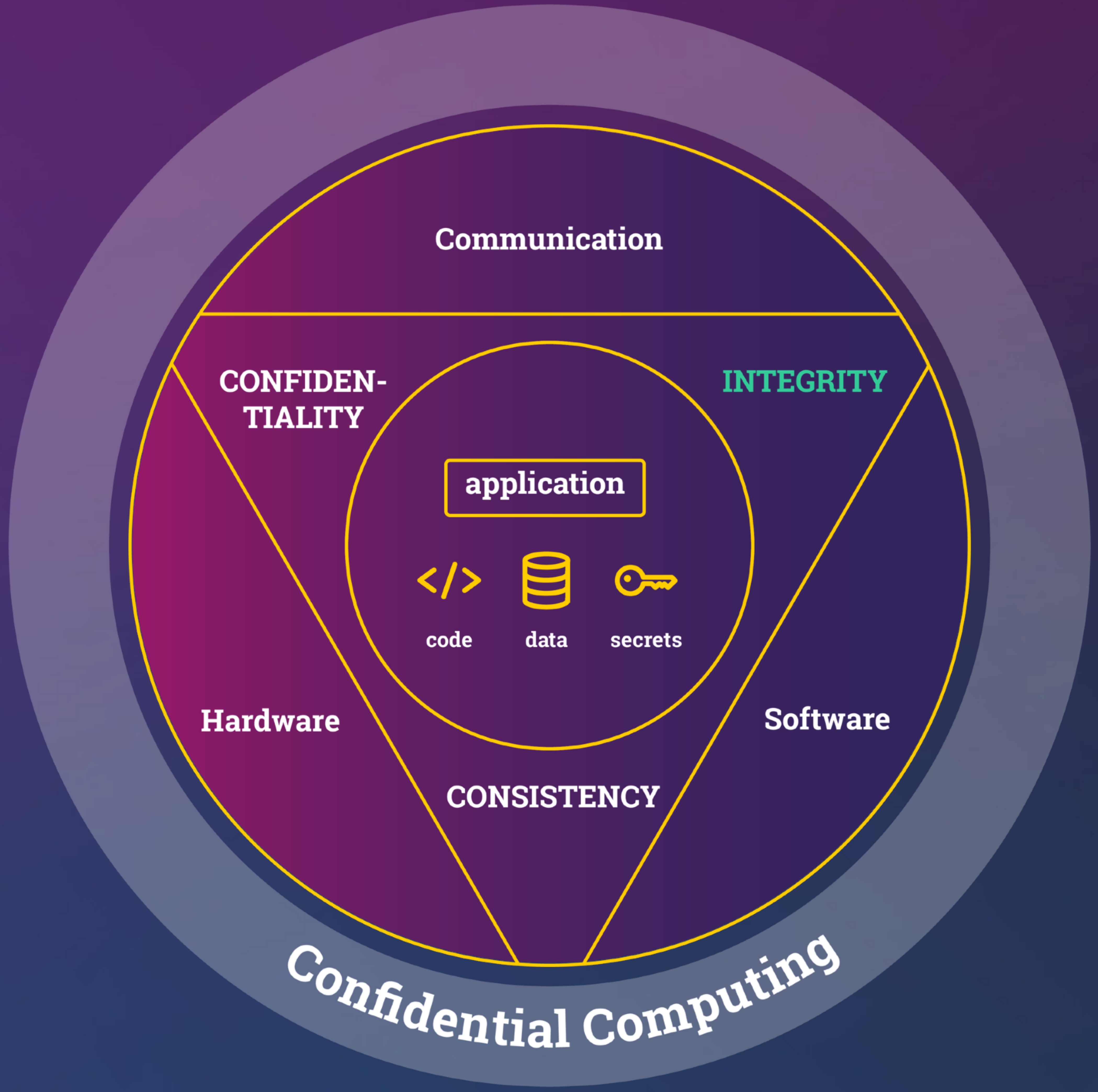
„application-oriented security“

Confidentiality



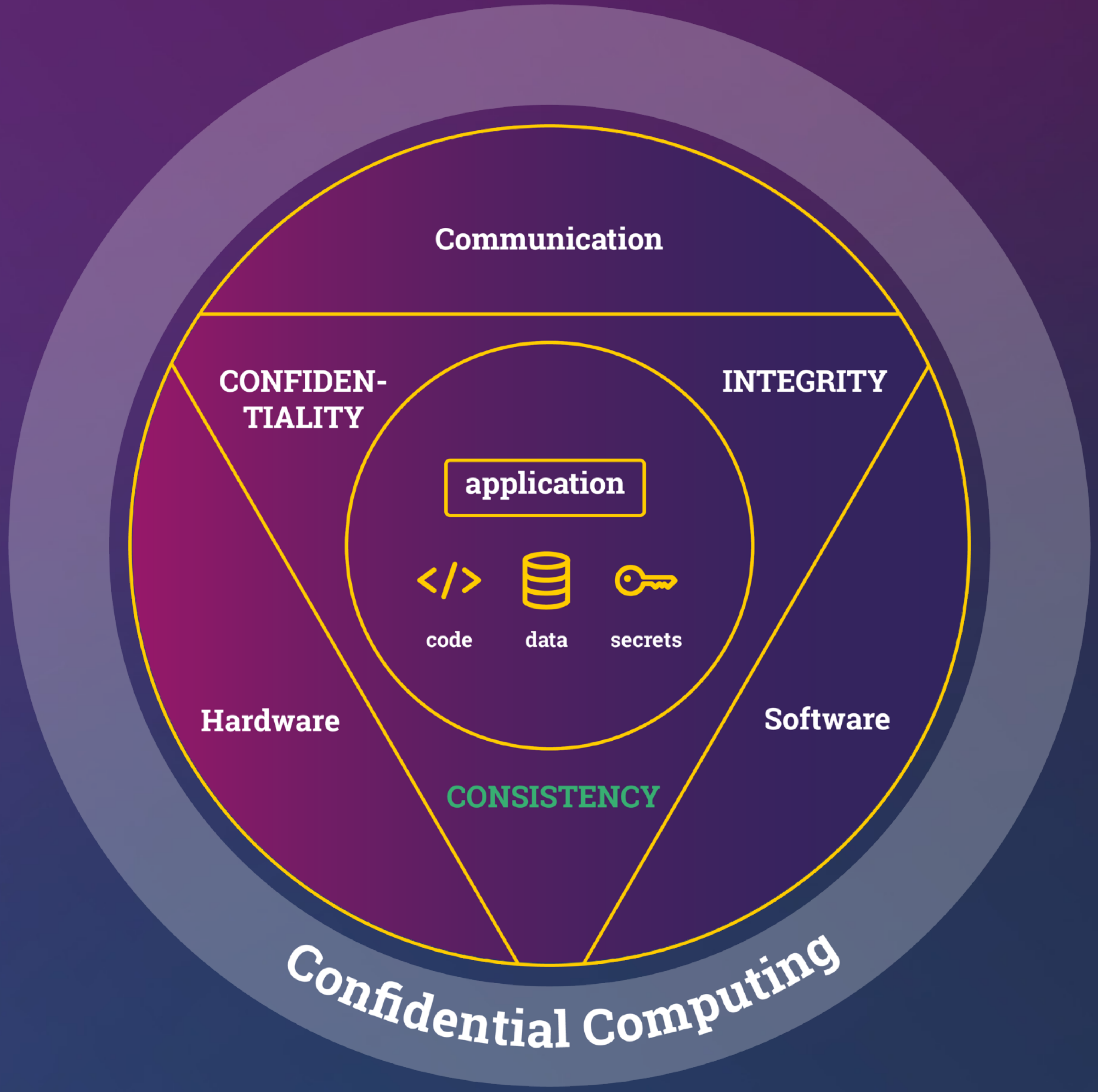
Confidentiality *is the property, that information (data, code, secrets) is not made available or disclosed to* ***unauthorized*** *individuals, entities, or processes.*

Integrity



Integrity *means that information (i.e., data, code, secrets) cannot be modified in an* ***unauthorized*** *or* ***undetected*** *manner.*

Consistency



Consistency *means that one always reads the latest information (i.e., data, code, secrets) written by an* ***authorized*** *entity.*

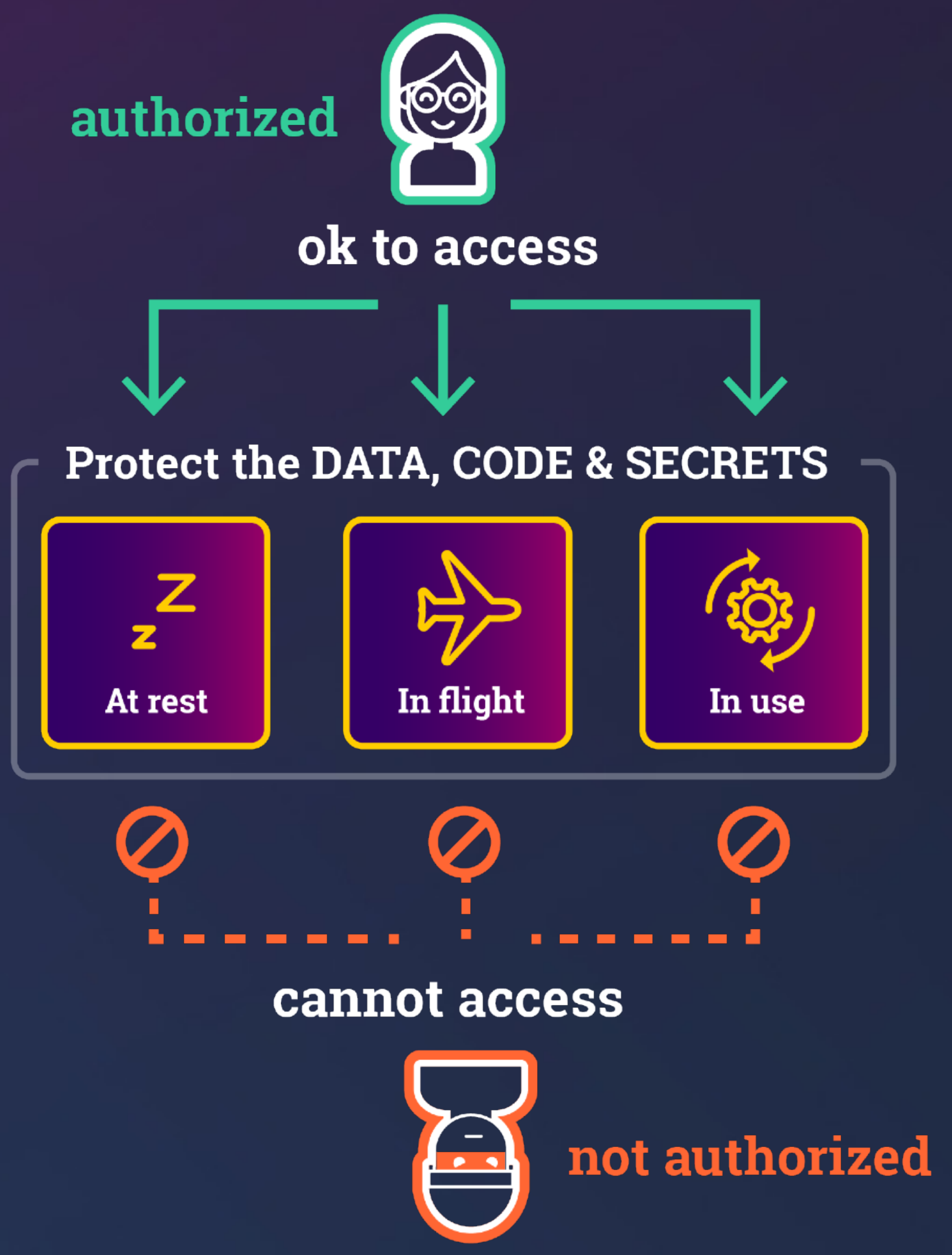
Detect if an adversary would provide an old copies (which are correctly encrypted but that have been updated).

PROBLEM:

How to define who is authorized
if one cannot trust the operating / hypervisor / … system?

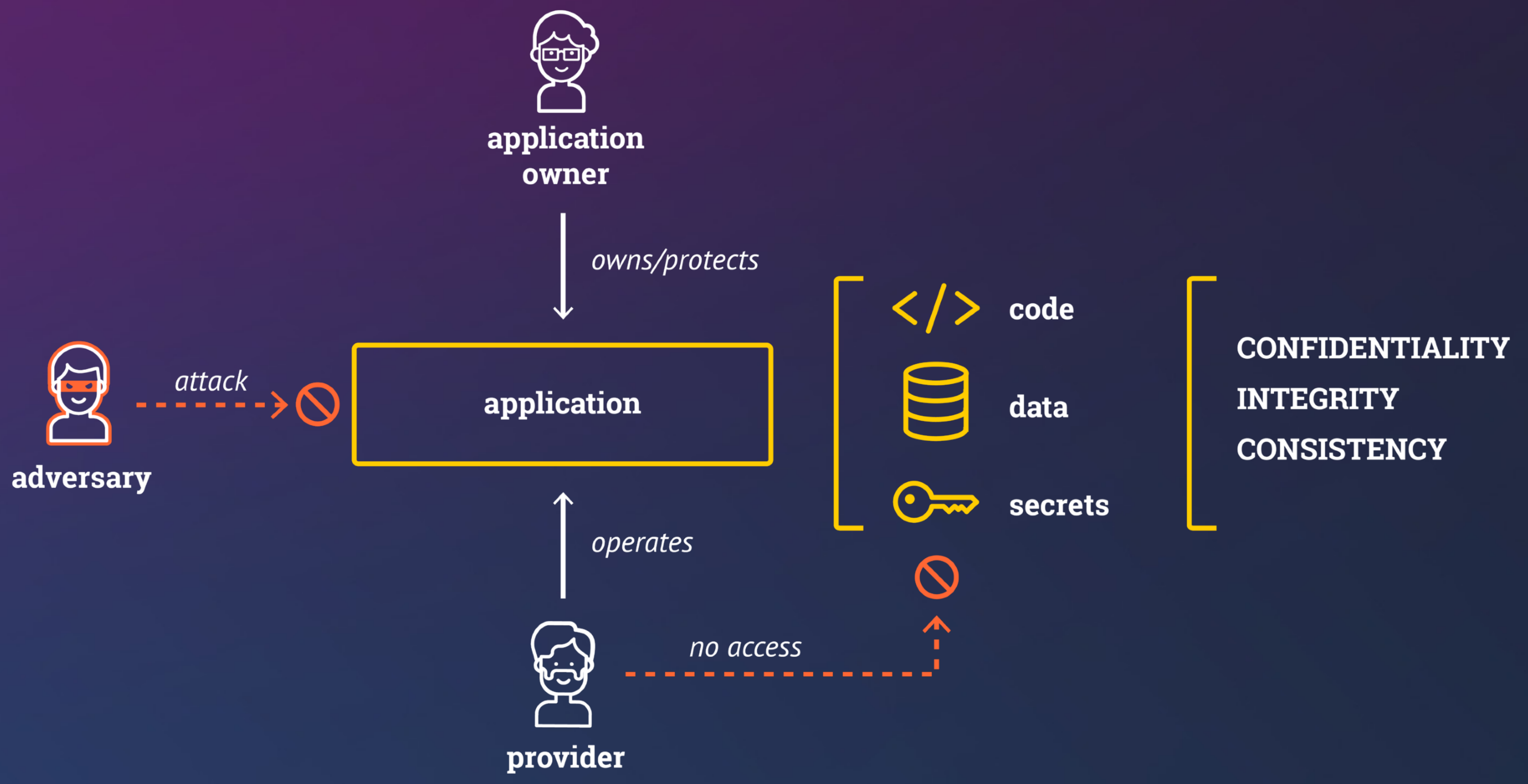
Protection Goals of Confidential Computing

- **Confidentiality:**
  - only authorized users/programs can read
- **Integrity:**
  - only authorized users/programs can update
- **Consistency**
  - always accessing the last version



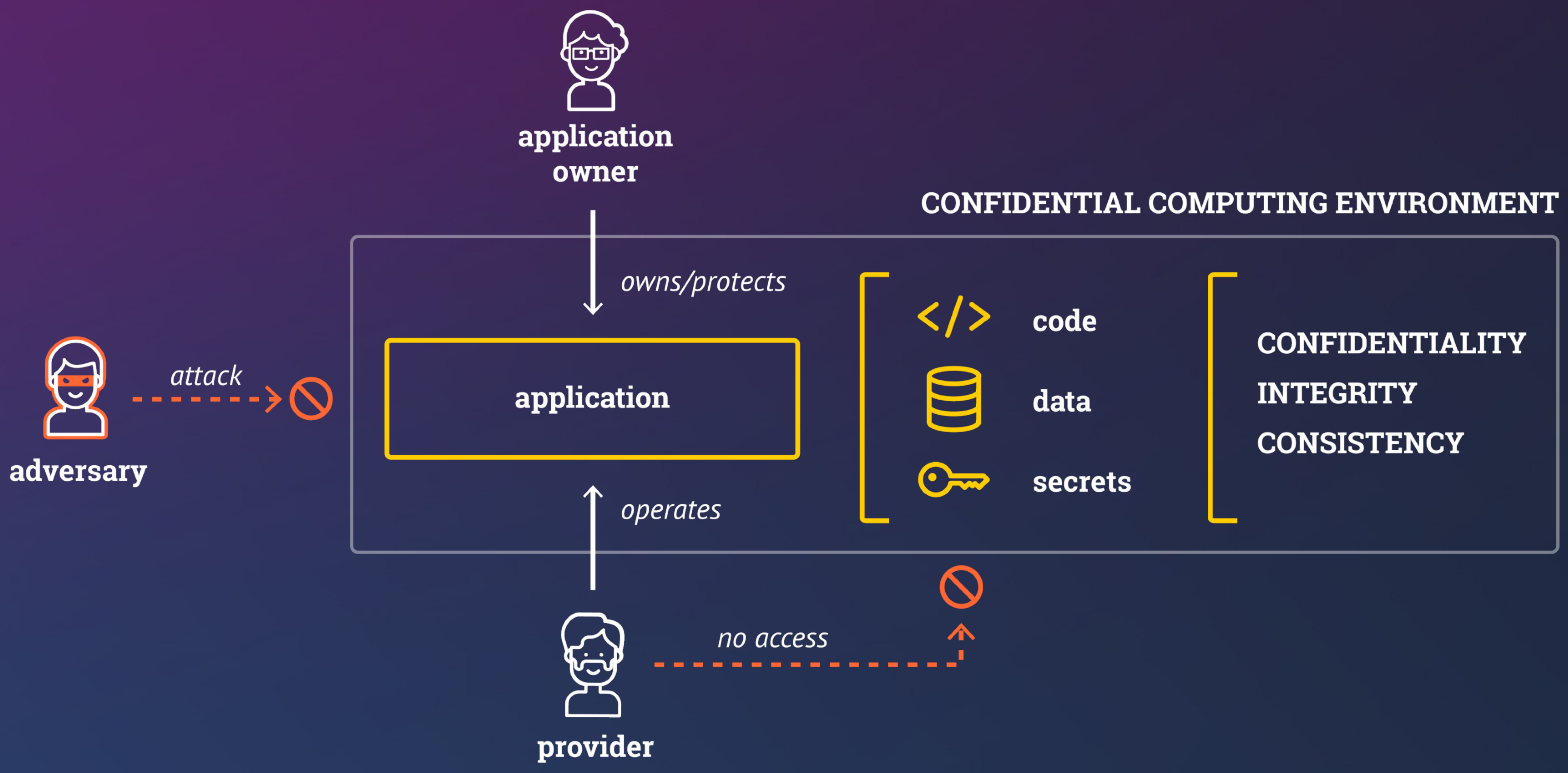
who is authorized?

Protect against Adversaries and Cloud Provider!

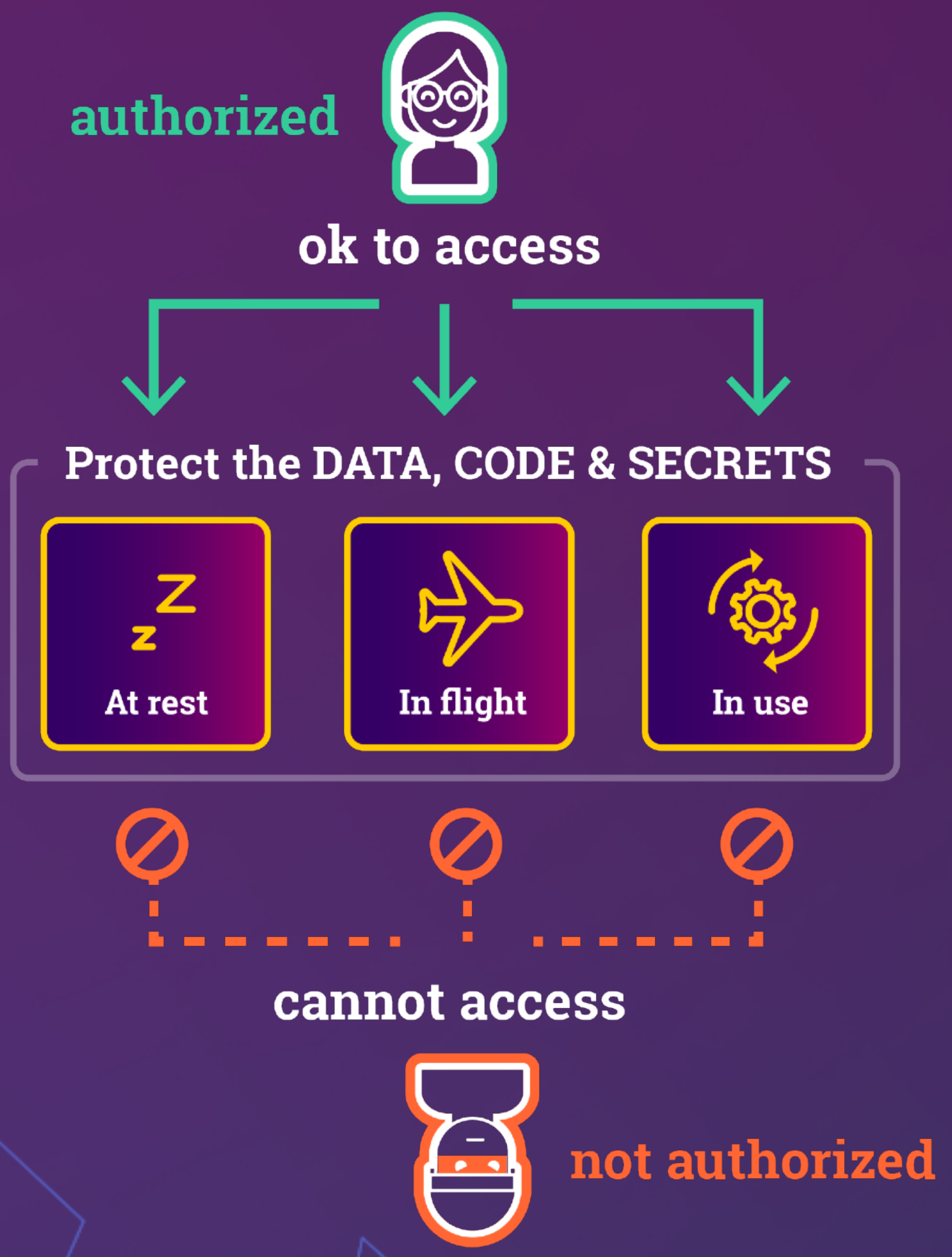


„application-oriented security“

Approach: Confidential Computing



„application-oriented security“



Protection in Use

- i.e., in main memory -

PROBLEM:

How to ensure confidentiality, integrity and consistency if adversary has root and hardware access?

PROBLEM:

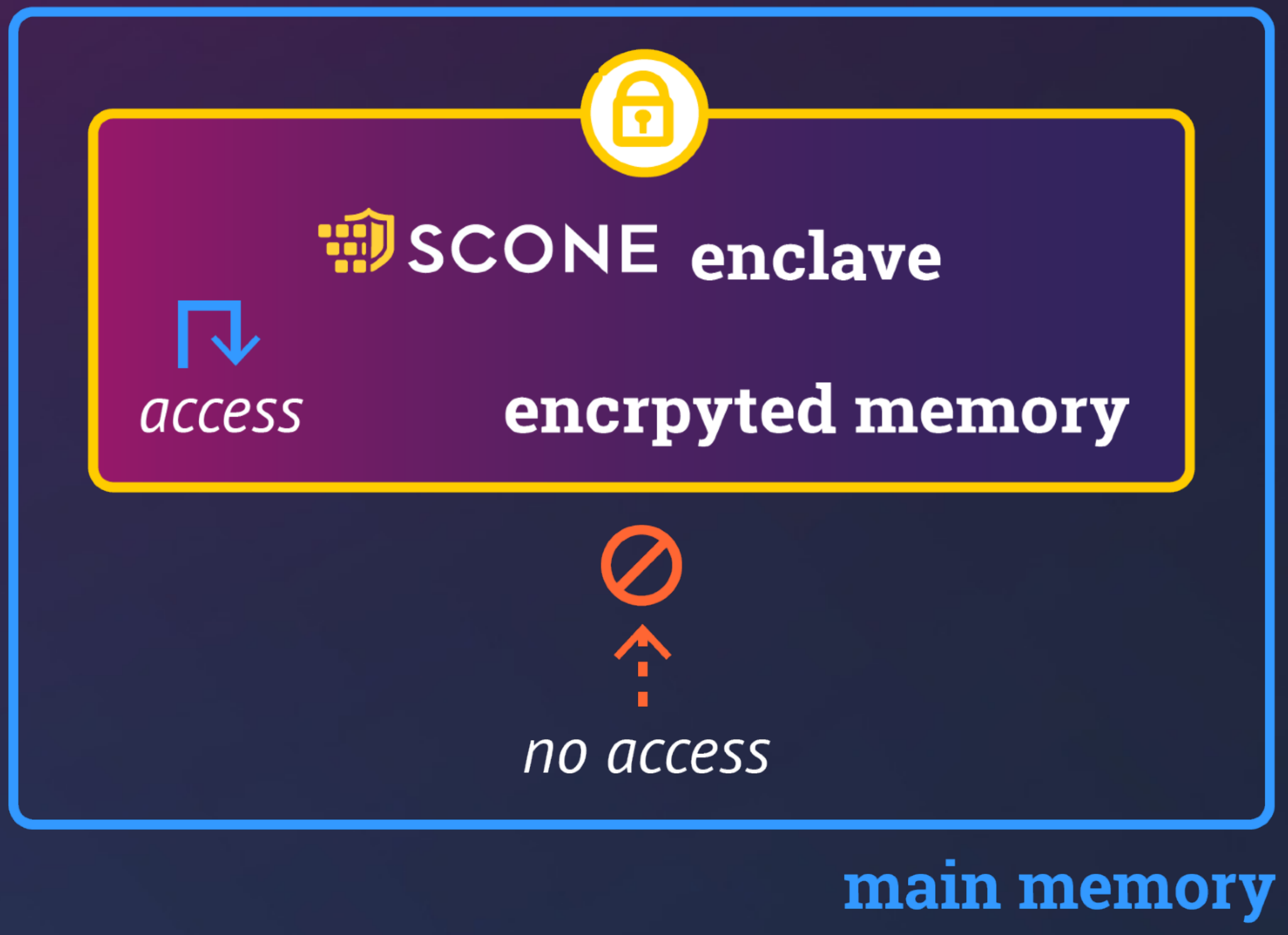
How to ensure confidentiality, integrity and consistency if adversary has root and hardware access?

Approach

Run applications in Trusted Execution Environments (TEE) without access by root users!

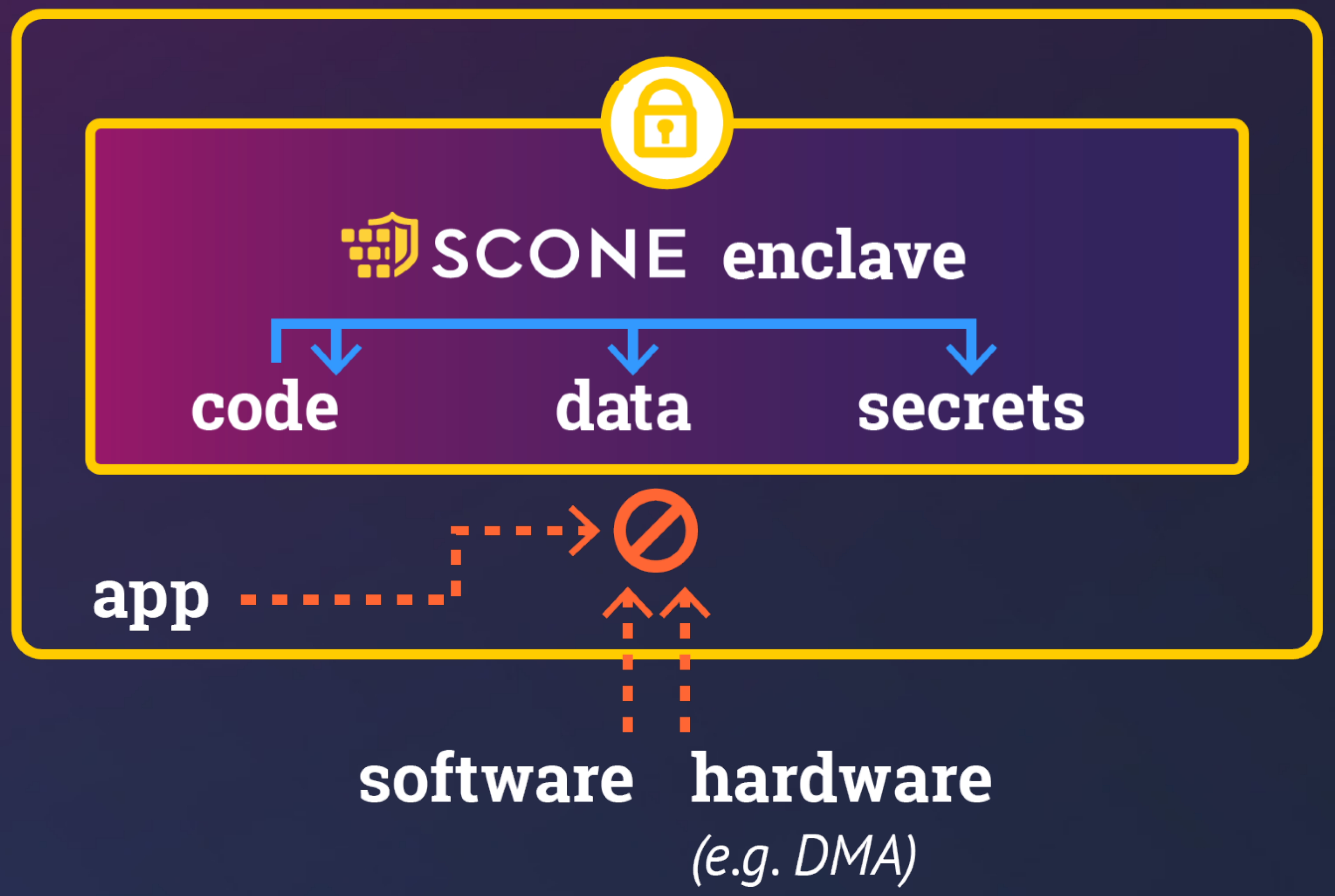
Trusted Execution Environment (TEE)

- Enclaves are a TEE:
  - CPU instruction set extension
  - service runs inside an encrypted memory region, aka, **enclave**
    - **random key** — generated by CPU, or
    - **external key** — given by external entity - **can we trust this key**?
  - only code running inside of enclave can see content
- **Different granularities:**
  - **enclave**: process runs inside encrypted memory (Intel SGX)
  - **encrypted VM**: a whole VM (AMD SEV, ARM Realms, Intel TDX)
    - can be used to implement enclaves



Enclaves

- **Protect data/code/secrets in use** (i.e, in main memory):
  - run application code in encrypted memory region (aka **enclave**)
  - only code in enclave can access region

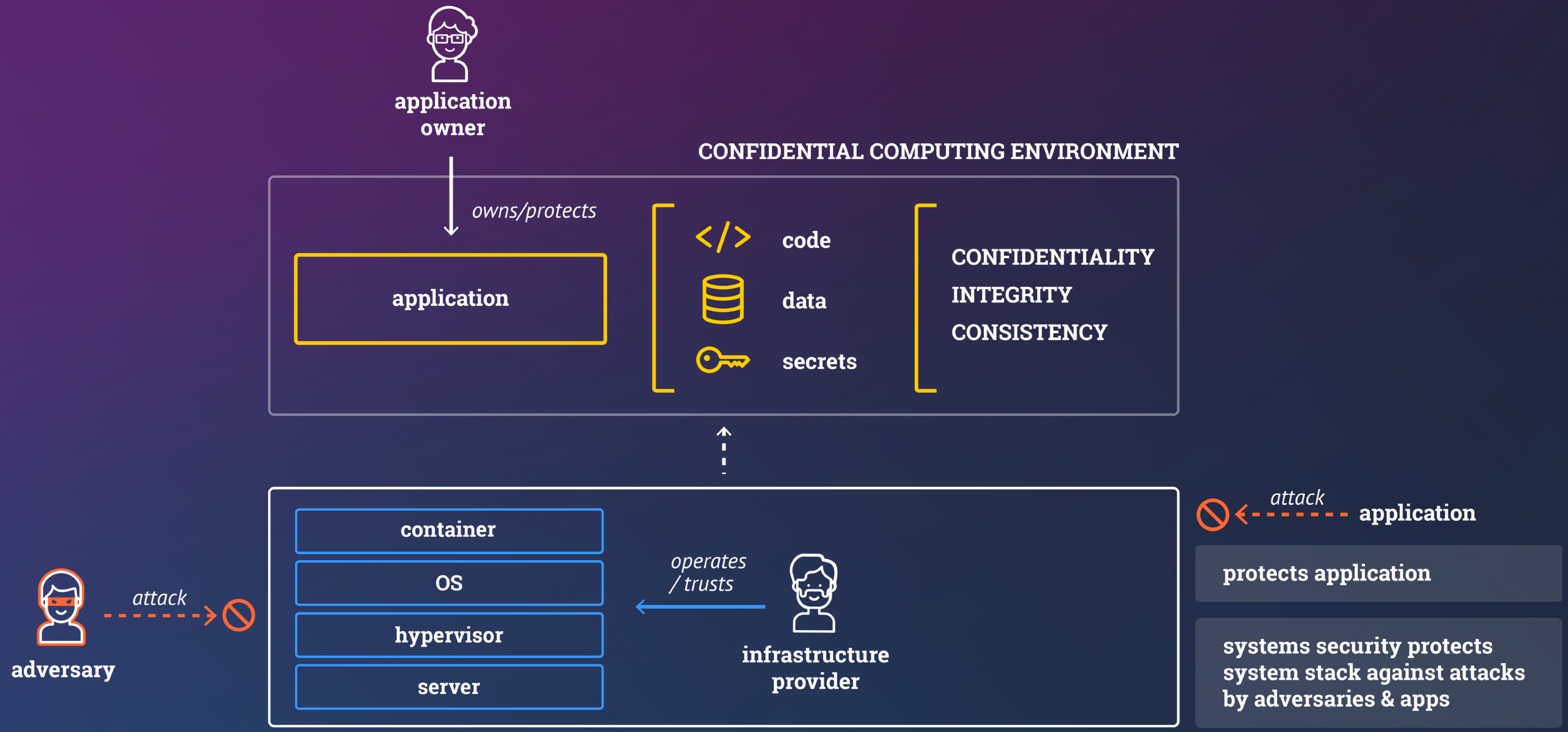


CPU extension: instructions to create enclave

Protecting Data in Flight: SCONE Network Shield

Protecting Data at REST: SCONE Fileshield

Complementary: Systems Security + Confidential Computing



Availability

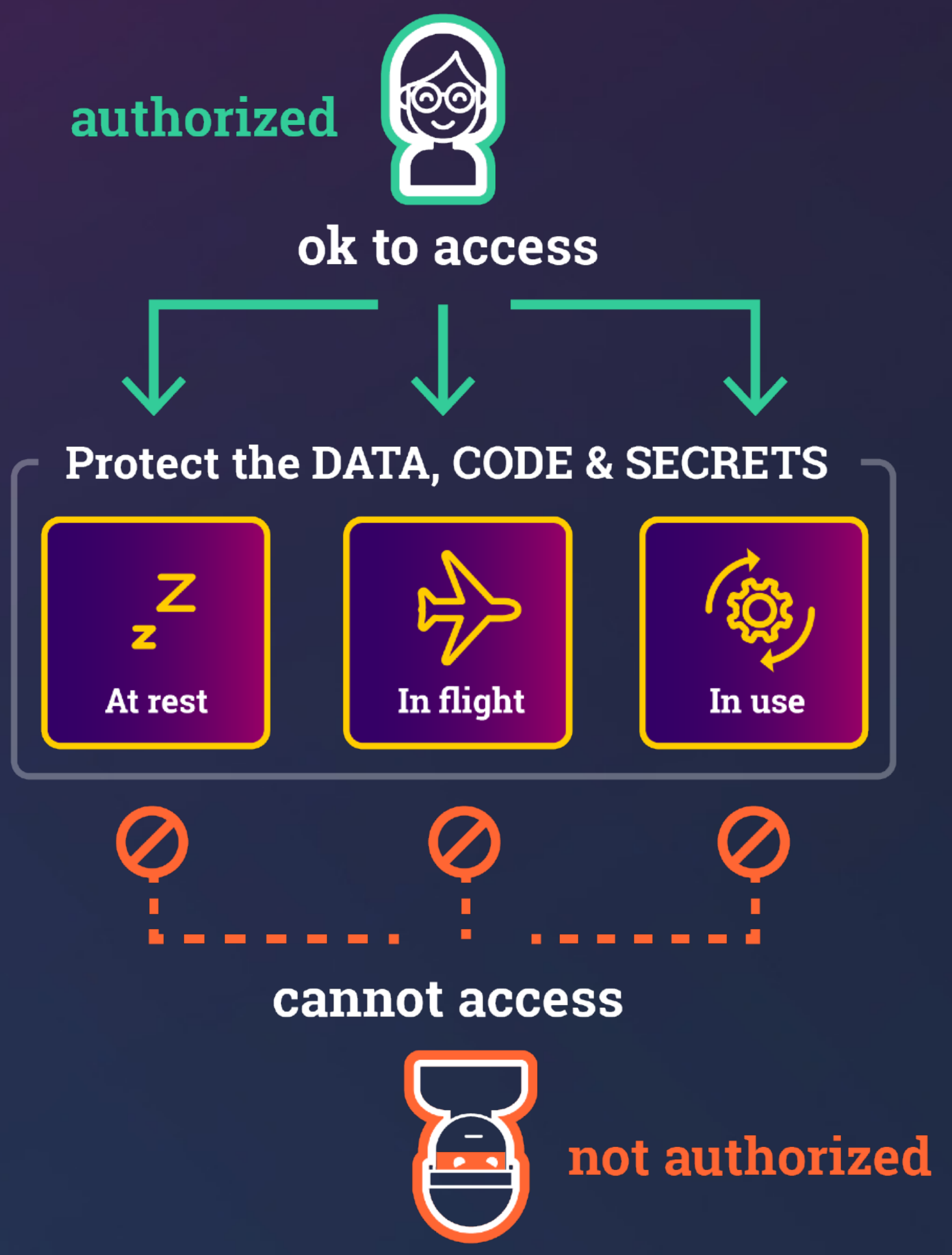
- **Definition**:
    - The probability that a **system**/ **application** / **service** will operate satisfactorily at a given point in time when
- Examples:
    - 99.99% (4 nines availability)
    - 99.999% (5 nines availability)
    - 99.9999% (6 nines availability)

Durability

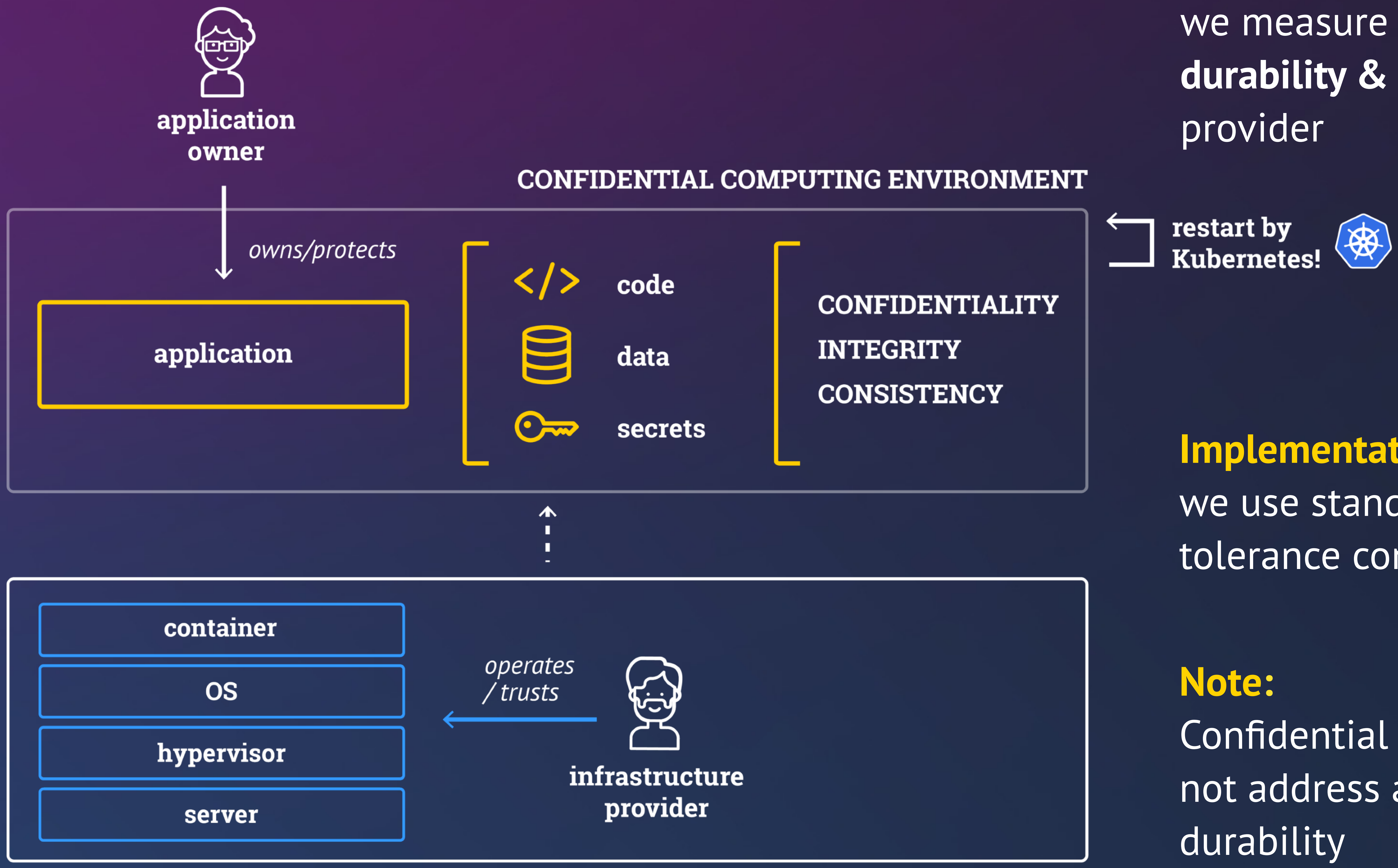
- **Definition**:
  - The probability that a **data item** will be accessible after one year
- **Examples**:
  - 99.999999999% (11 nines durability)

Availability and Durability

- **Protection goals NOT addressed by confidential compute:**
  - **Availability**: the probability that information is available when it is needed
  - **Durability**: the probability that information will survive for one year
- **Approach**:
  - address this via SLA (Service Level Agreements)



Availability



General Approach:
we measure **availability** and **durability &** define SLA with provider

Implementation:
we use standard fault-tolerance concepts

Note:
Confidential Computing does not address availability nor durability

(SLA = Service Level Agreement)

Example

What information to protect?

- Protection of
  - **Code**, e.g., modern AI programs written in Python
  - **Data**, e.g., training data to create AI models
  - **Secrets**, e.g., key used to encrypt database

What information to protect?

- Protection of
  - **Code**, e.g., modern AI programs written in Python
  - **Data**, e.g., training data to create AI models
  - **Secrets**, e.g., key used to encrypt database
- **Example**:
  - *MariaDB* supports encryption of database
  - encryption key is stored in configuration file
  - configuration file protected via access control:
    - i.e., can be read and written by MariaDB (user) as well as any root (=privileged) user

Is Operating System access control sufficient?

- Sufficient if we define that
  - only **authorized user** can become root users

Is access control sufficient?

- Sufficient if we define that
  - only **authorized user** can become root users
- What about
  - **adversary** gaining root access (e.g., stealing credentials)?
  - **authorized** user laid off -> could become an **adversary**?

Is access control sufficient?

- Sufficient if we define that
  - only **authorized user** can become root users
- What about
  - **adversary** gaining root access (e.g., stealing credentials)?
  - **authorized** user laid off -> could become an **adversary**?
- **Approach**:
  - Define **threat model** that defines the power of the adversary

Confidential Computing

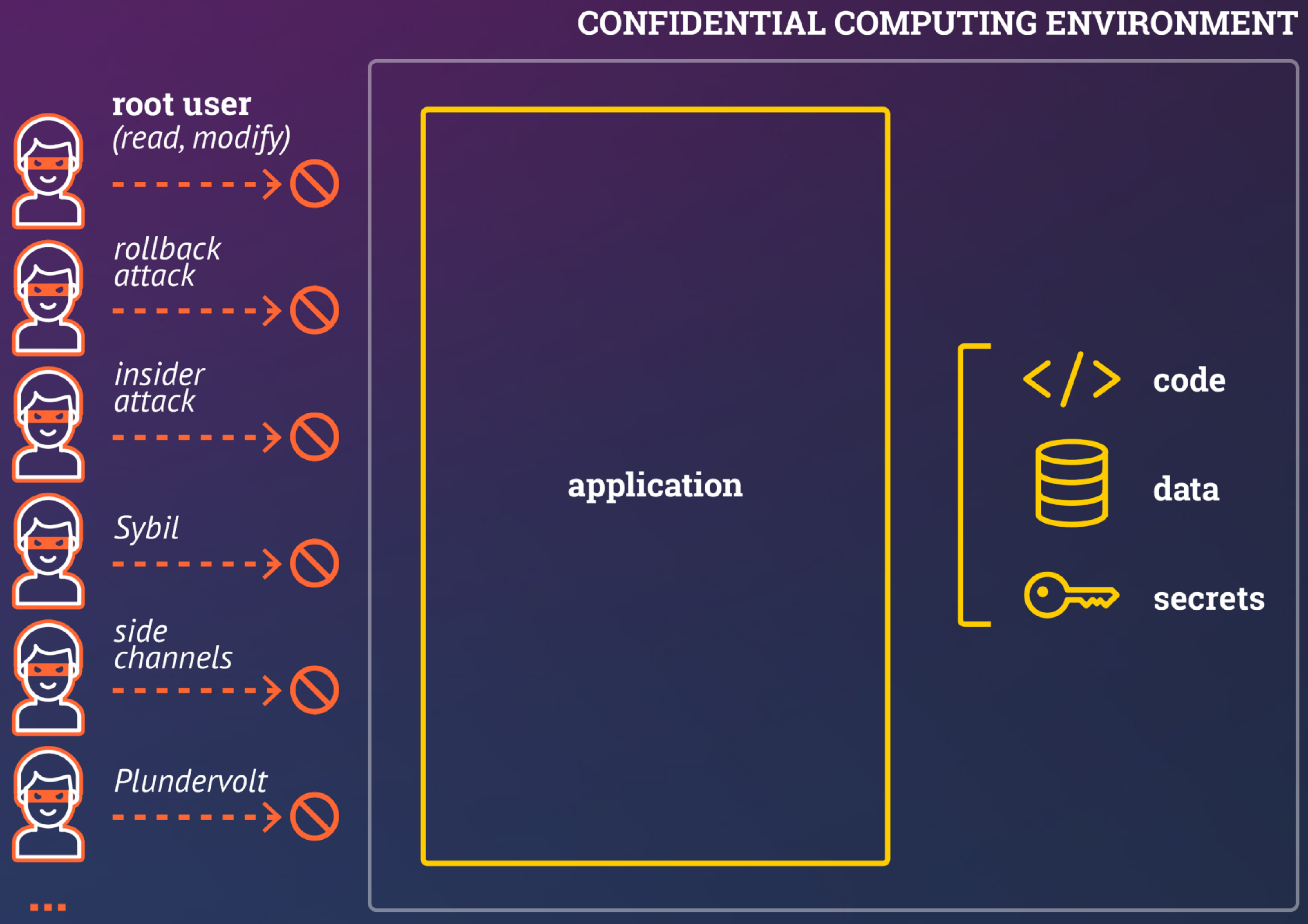
with SCONE
- Threat Model -

Christof Fetzer
https://sconedocs.github.io

Threat Model

- What Assets? Adversaries? Attack points? -

Lots of attacks known and possible…



„application-oriented security"

Threat Model

- A1) **Unprivileged Software Adversary**
- A2) **System Software Adversary**
- A3) **Startup Code/SMM Software Adversary**
- A4) **Network Adversary**
- A5) **Software Side-Channel/Covert-Channel Adversary**
- A6) **Simple Hardware Adversary**
- *A7) **Roll-Back State Adversary***
- *A8) **CVE Adversary***

https://www.intel.com/content/www/us/en/security/security-practices/secure-development-practices/threat-modeling.html

Unprivileged Adversary

- ***(A1) We assume that an adversary might be able to run user code on the same server***
- In a **cloud setting**:
  - an adversary can run applications by creating an account
- **Difficulties for adversary:**
  - the adversary's code might not run on the same server (1000s of machines)
  - the defender might rent a whole server (bare metal server) - no co-location with other „tenants“
    - **defender**: how to verify that no other application runs on the same server?
  - adversary's and defender's code are in different VMs (Virtual Machine, i.e., sandbox)
  - …

https://www.intel.com/content/www/us/en/security/security-practices/secure-development-practices/threat-modeling.html

System Adversary

- (A2) ***System Software Adversary: we assume that an adversary might have administrators' credentials with root access to the system software. Or, they might coerce admins into copying or modifying data or code both at the system and the application level.***

- Note:

  - *adversary can read all memory (e.g., by „attaching“ to process)*

  - *adversary can modify memory, modify system call arguments / return values, can read/modify arbitrary files, can read/modify network traffic,…*

Why assume System Software Adversary?

- **Reasons**:
  - **Legal**:
    - cloud provider / cloud employee might be legally required to provide access to the data
  - **Liability**:
    - too expensive to err on the threat model
  - **Limits of access control:**
    - How do we know that we can trust individual user?
  - **Software complexity**: cannot assume that software is correct (see **Defender's dilemma**)
  - **Hardware complexity**: cannot assume that hardware is correct (see **BMC**, **firmware**, …)
  - …

Adversaries

- A3) **Startup Code/SMM Software Adversary.** We assume that an adversary can modify the code initially booted on the server and change the system management code. **This adversary might also try to access the machine via the BMC.**

- A4) **Network Adversary**. We assume that an adversary with access to the network can try to communicate with app to gain access or trigger bugs in app.

- A5) **Software Side-Channel/Covert-Channel Adversary**: an adversary could try to extract secrets and keys from app using side channels that circumvent the protections of the enclaves.

Adversaries

- A6) Simple Hardware Adversary
  - A6.1) The adversary can remove DRAMs, has tools for hardware debugging, and can listen to buses.
  - A6.2) The adversary gets hold of decommissioned hardware. The adversary recovers a CPU, TPM, and database when the hardware or service is decommissioned.
- **A7) Roll-Back State Adversary:**
  - A7.1) The adversary can **roll back the disk state**. The disk state might contain an encrypted database with keys that have been updated already.
  - A7.2) The adversary can **roll back the CPU firmware** or the app code. The adversary can install an old version of the CPU firmware to exploit known bugs in enclave implementation. An adversary could use this to extract information from, say, an encrypted key database.
  - A7.3) The adversary can freeze the state of an existing app. This is to keep access to an earlier version of the database.

Adversaries

- A8) CVE Adversary

- The adversary knows all components of an application, i.e., all dependencies and their versions of each of the application services are known. The adversary knows all the vulnerabilities that affect these components and will exploit these vulnerabilities to attack the application.

Threats Not Covered

- A1) **Unprivileged Software Adversary**
- A2) **System Software Adversary**
- A3) **Startup Code/SMM Software Adversary**
- A4) **Network Adversary**
- A5) **Software Side-Channel/Covert-Channel Adversary**
- A6) **Simple Hardware Adversary**
- *A7) **Roll-Back State Adversary***
- *A8) **CVE Adversary***
- A9) **Skilled Hardware Adversary**
- A10) **Hardware Reverse Engineer Adversary**
- A11) **Authorized Adversary**

https://www.intel.com/content/www/us/en/security/security-practices/secure-development-practices/threat-modeling.html

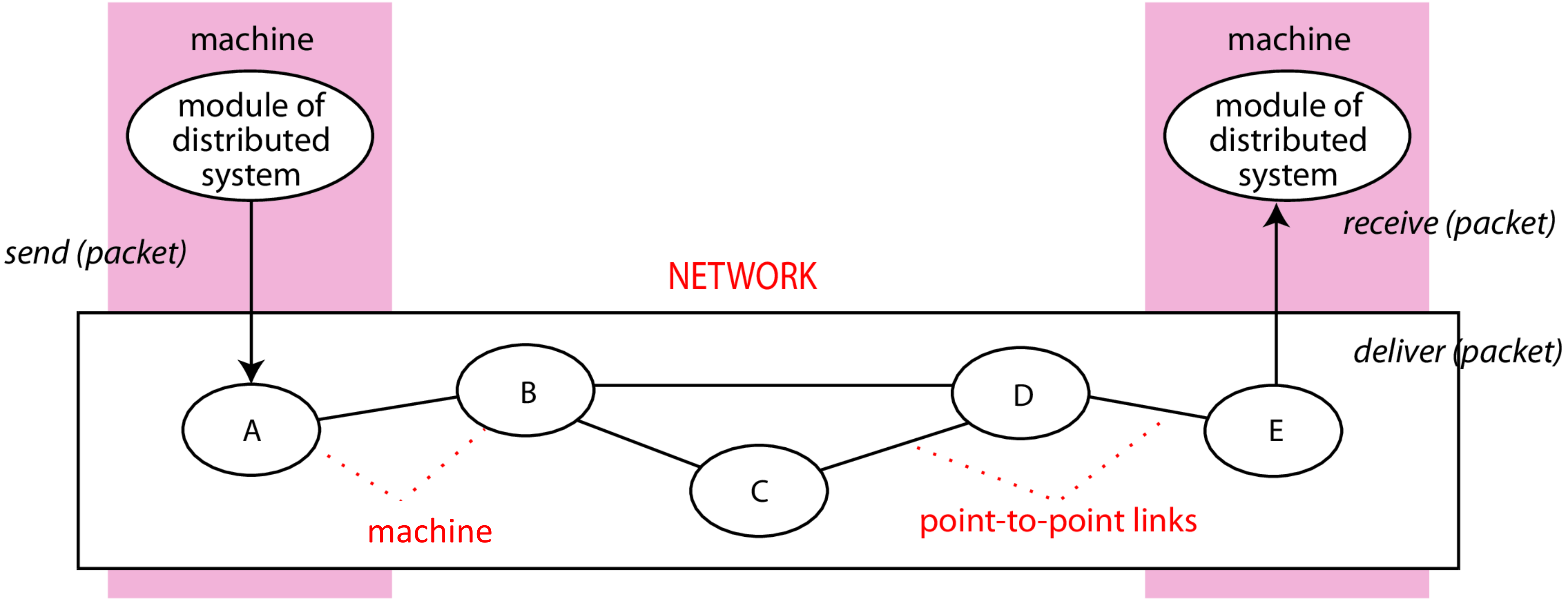
AI-SPRINT
AI IN SECURE PRIVACY-PRESERVING COMPUTING CONTINUUM

Network Security at the Edge

Introduction and Attacks

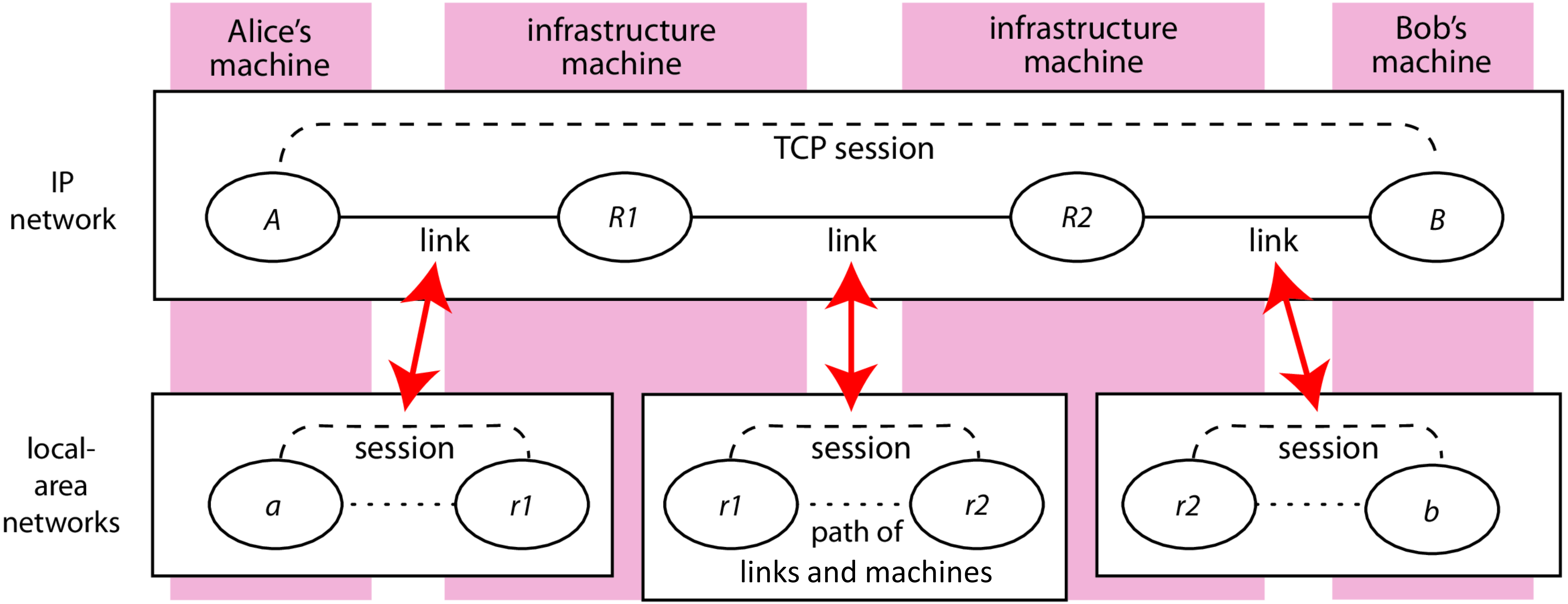
AI-SPRINT project has received funding from the European Union Horizoon 2020 research and innovation programme under Grant Agreement **No. 101016577.**

Model of a Single Network



Source: Zave, P., and J. Rexford. 2020. "Patterns and Interactions in Network Security." *ACM Computing Surveys (CSUR)*.

Composition of Networks



Source: Zave, P., and J. Rexford. 2020. "Patterns and Interactions in Network Security." *ACM Computing Surveys (CSUR)*.

What is an Attacker?

- Malicious User: a user machine that has access to the network (rightfully or maliciously) and behave **dishonestly** or erroneously

- Malicious Network: legitimate machines **taken over** by some attacker, or external machines where traffic has been **maliciously steered**

Flooding Attack

- The attacker sends **floods of packets** to exhaust resources (CPU, memory, bandwidth)
- Leverages some form of **amplification**:
  - Botnets & Distributed Denial-of-Service
  - Asymmetric resource consumption
  - Reflection and amplification by an intermediate machine

- More than simple vandalism:
  - Attack to DNS impacts **many users**
  - Massive IoT attacks can **disrupt** businesses, industries, or critical systems
  - Attack to infrastructure **shifts traffic** to unsafe network

Subversion Attack

- Take **control of the victim**’s actions:
  - spreading malware,
  - information gathering,
  - route hijacking,
  - spam

- Requires two-way communications (differently from flooding)

Policy Violations

- Administrations (or governements) can **ban** certain communications
  - criminals
  - saboteurs
  - minors
  - industrial property, industry secrets

Spying and Tampering

- Attacks the **confidentiality or integrity** of a private communication, makes the network not transparent
  - surveillance and censorship
  - information gathering about users
  - communications hijacking (e.g. insert ads or alter search results)

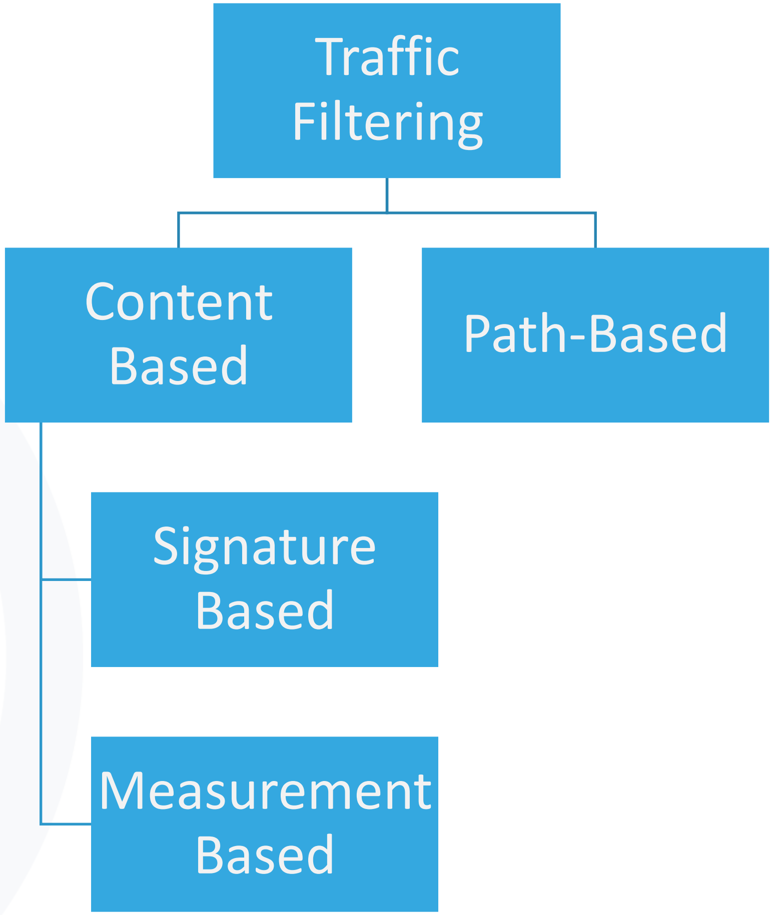
Classification of Attacks

Attack	Attackers	Victims	Defense Mechanism
Flooding	Users	Network, Users	traffic filtering resource allocation
Subversion	Users	Network, Users	traffic filtering cryptographic protocols
Policy Violation	Users	Network	traffic filtering
Spying and Tampering	Network, Users	Users	cryptographic protocols overlays

- Executed at the endpoints to provide
  - data confidentiality: the attacker cannot read the packets in a session
  - data integrity: the attacker cannot insert, modify, replay packets in a session
  - endpoint authentication: the attacker cannot impersonate a legitimate user

Protocol	Endpoint Authentication	Data Confidentiality	Data Integrity
TLS / HTTPS	Optional for client Mandatory for server	Only payload, not source and destination addresses	Yes, also protection against replay, deletion, reordering
Signal Protocol (e.g. WhatsApp)	Yes, mutual	Only payload, not source and destination addresses	Yes, also protection against replay, deletion, reordering

- Network ensures that traffic passes through traffic filters
- The filter looks for evidence of attack (flooding, subversion, policy) and drops or accepts the packets



Traffic filtering devices

	Router	Stateful Firewall	Intrusion Detection System	Intrusion Prevention System
Filtering Criteria	Packet headers	Packet headers; table of ongoing sessions	Packet haders and payloads	Packet haders and payloads
Require Session Tracking?	No	yes	yes	yes
Actions Taken	Drop	Drop, limit rate	Alert, record	Drop, limit rate, normalize

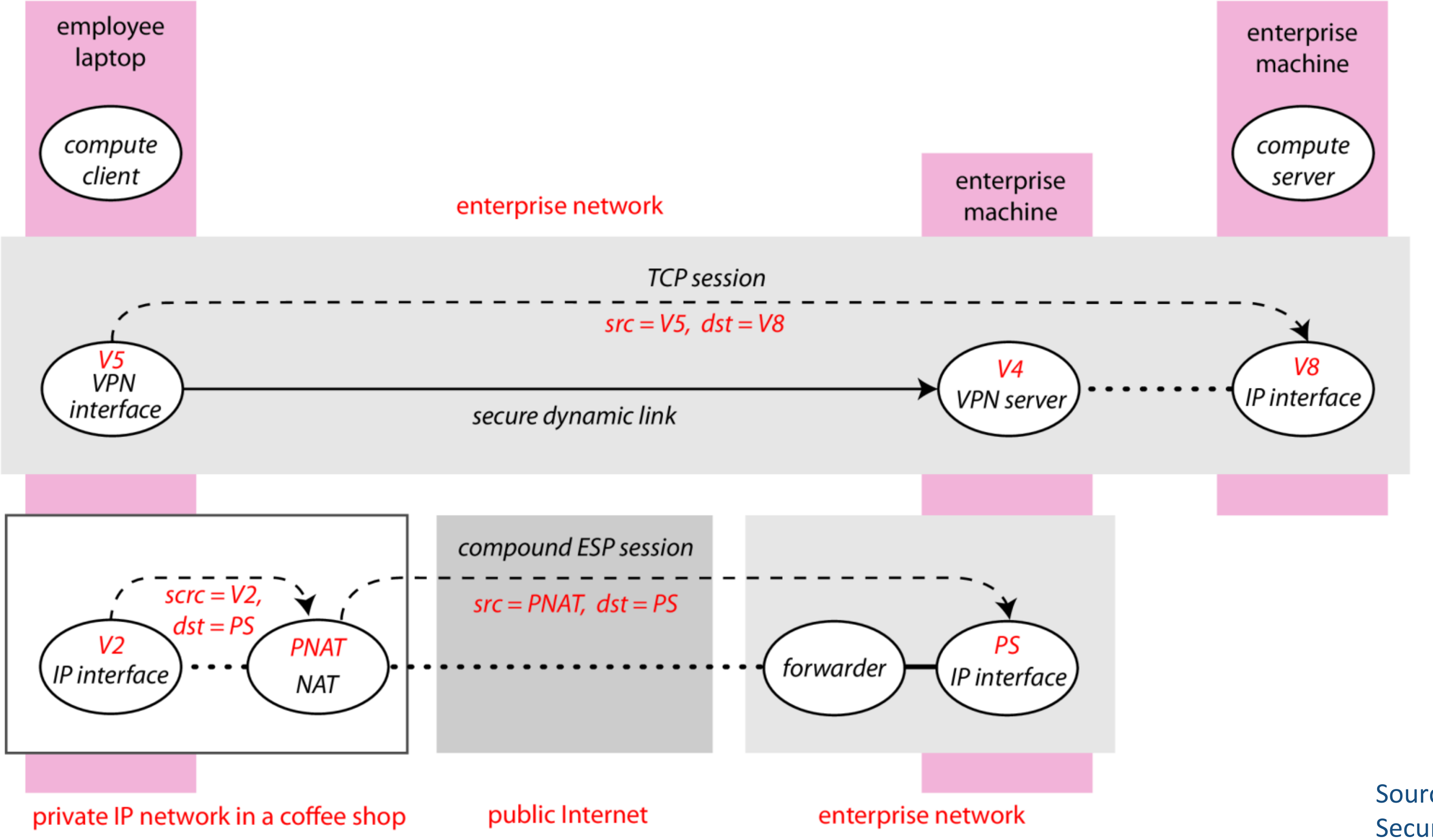
Resource Allocation

- Cloud computing makes it easy to scale servers and to scale traffic filters *(server-centric defense).*

- When servers are scaled out, the network must distribute the load

Network Overlays

- Cryptographic protocols have drawbacks:
  - do not protect packet headers
  - prevent traffic filtering on the payload.



Source: Zave, P., and J. Rexford. 2020. "Patterns and Interactions in Network Security." *ACM Computing Surveys (CSUR)*.

AI SPRINT
AI IN SECURE PRIVACY-PRESERVING COMPUTING CONTINUUM

Network Security at the Edge

5G and Multiaccess Edge Computing (MEC)

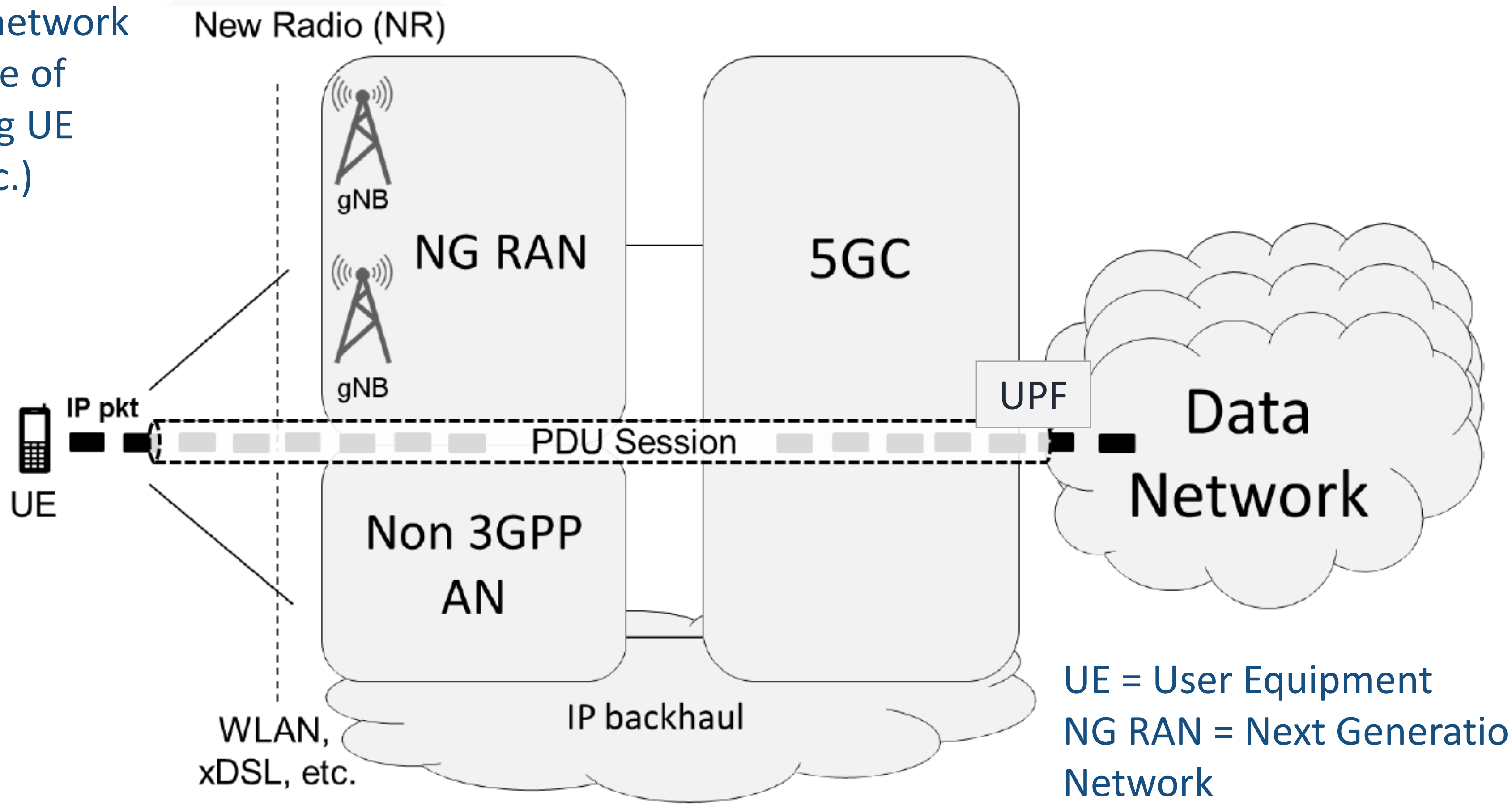
AI-SPRINT project has received funding from the European Union Horizoon 2020 research and innovation programme under Grant Agreement **No. 101016577.**

- Before 5G, the cellular network was centered on providing services to **people**, and the architecture was made of **physical nodes** running **proprietary software** on proprietary hardware, often in a single physical device.

- 5G introduced two innovations:
  - the **verticals**, i.e. services addressing various vertical markets (automotive, energy, healthcare, etc.)
  - **network softwarization**, by which network functions are virtualized and run on a cloud

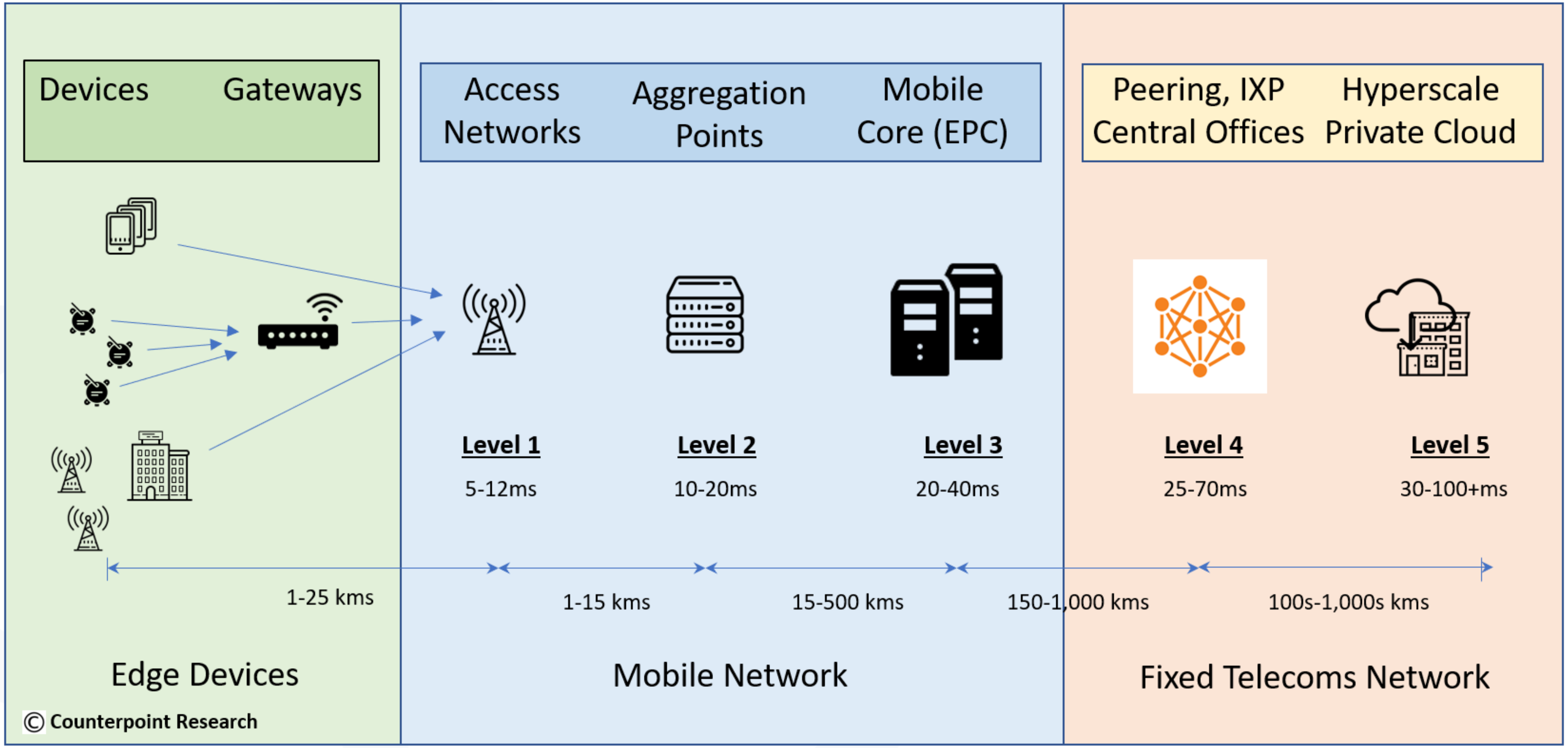
5G Network Architecture

A 5G network connects a UE to an external network through a sequence of tunnels (supporting UE mobility, states, etc.)

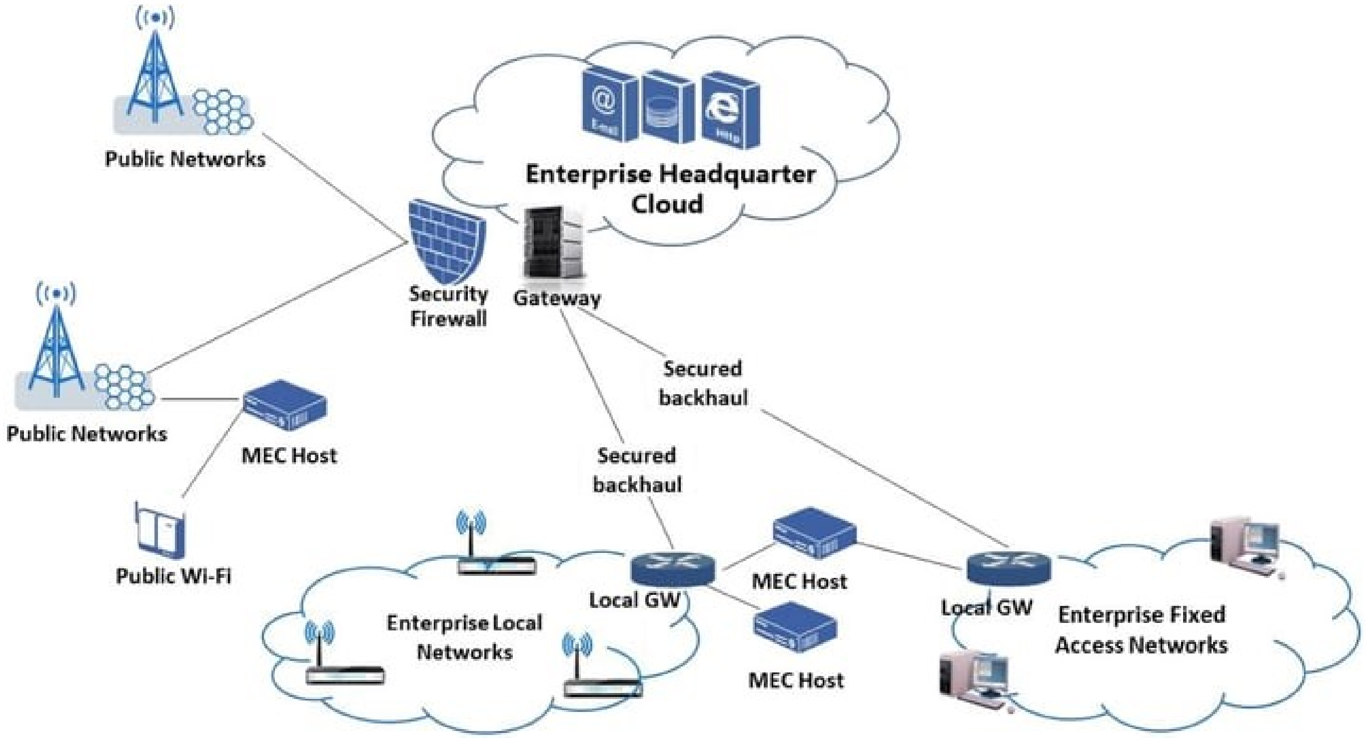


UE = User Equipment
NG RAN = Next Generation Radio Access Network
5GC = 5G Core
UPF = User Plane Function

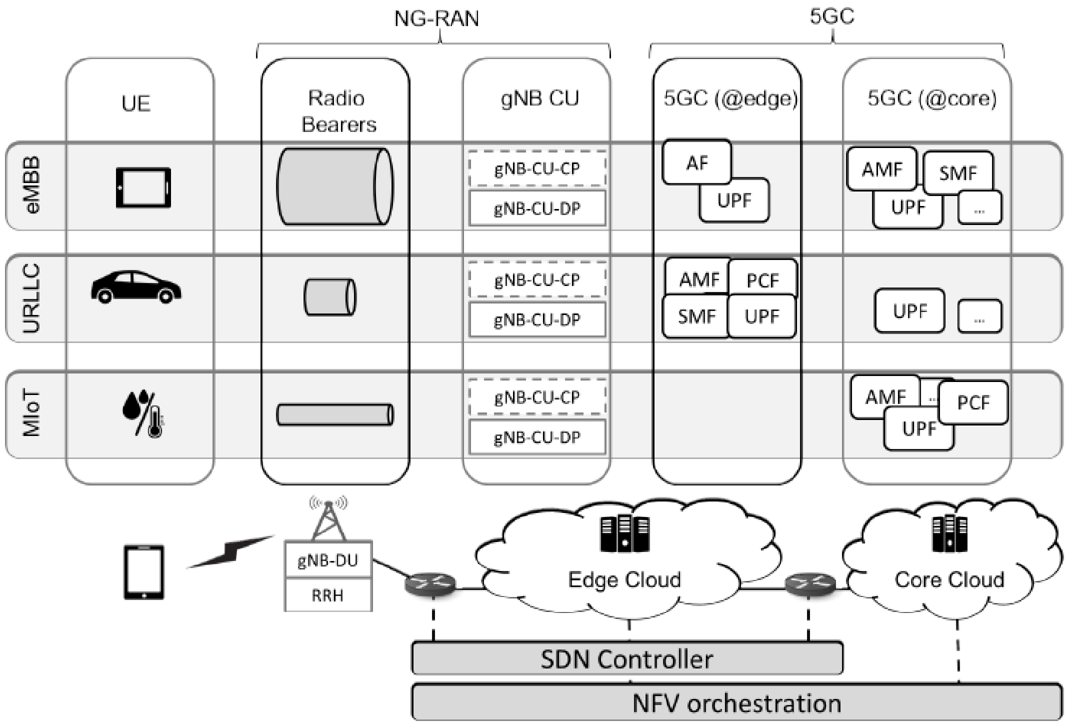
A Vision of the Edge

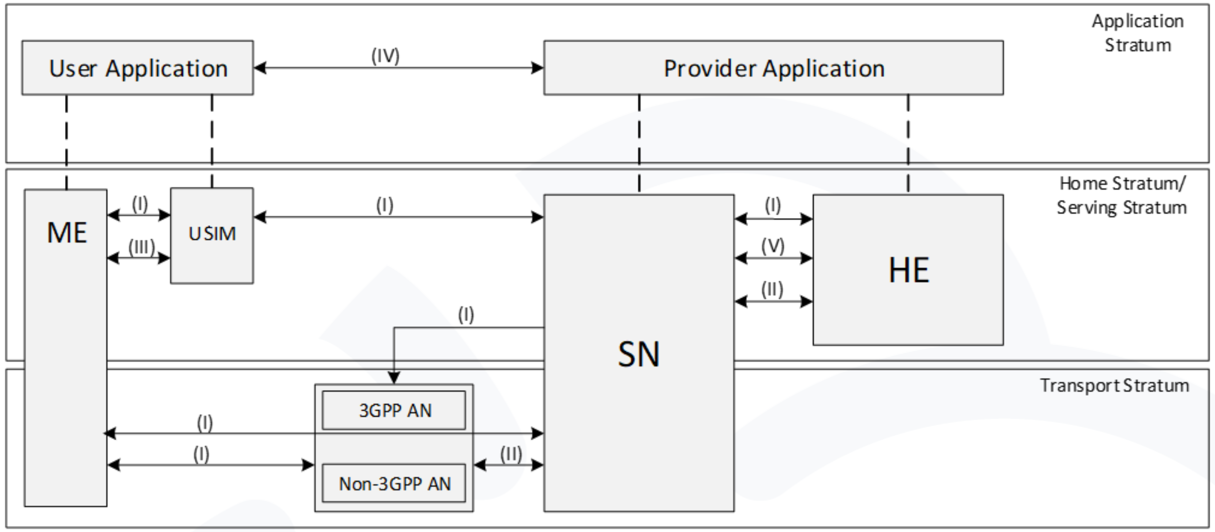


- An ETSI standard computing architecture
- Brings **cloud-native applications** at the edge
- Use cases:
    - **low latency** applications (augmented reality, connected cars, industrial control, etc)
    - **on-premises** data centers
- Enables private 5G deployments in which all or part of the 5G infrastructure is provided by the enterprise



- A Slice is a set of instances of network functions with isolated resources
- They are a kind of Network-as-a-Service offered to each of the verticals
- Standardized verticals:
  - eMBB: enhanced Mobile BroadBand
  - URLLC: Ultra-Reliable Low Latency Communications
  - MIoT: Massive IoT





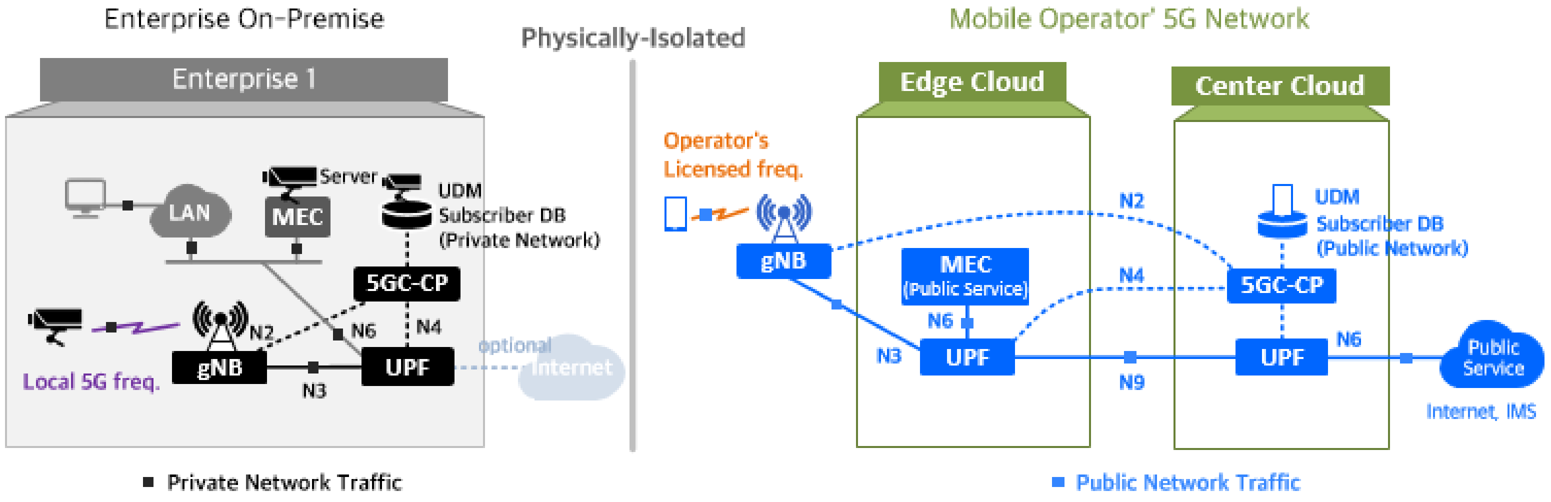
- Application Stratum: handles application security (not 5G-specific)
- Home/Service Stratum: handles subscriber authentication, device authentication, service authentication
- Transport Stratum: handles infrastructure security

ME = Mobile Equipment
SN = Serving Node
HE = Home Equipment

Threats to the Infrastructure:

- Rogue Base Stations & Man-in-the-Middle
- DoS/Flash Traffic on the Infrastructure and the Devices
- Virtualization security

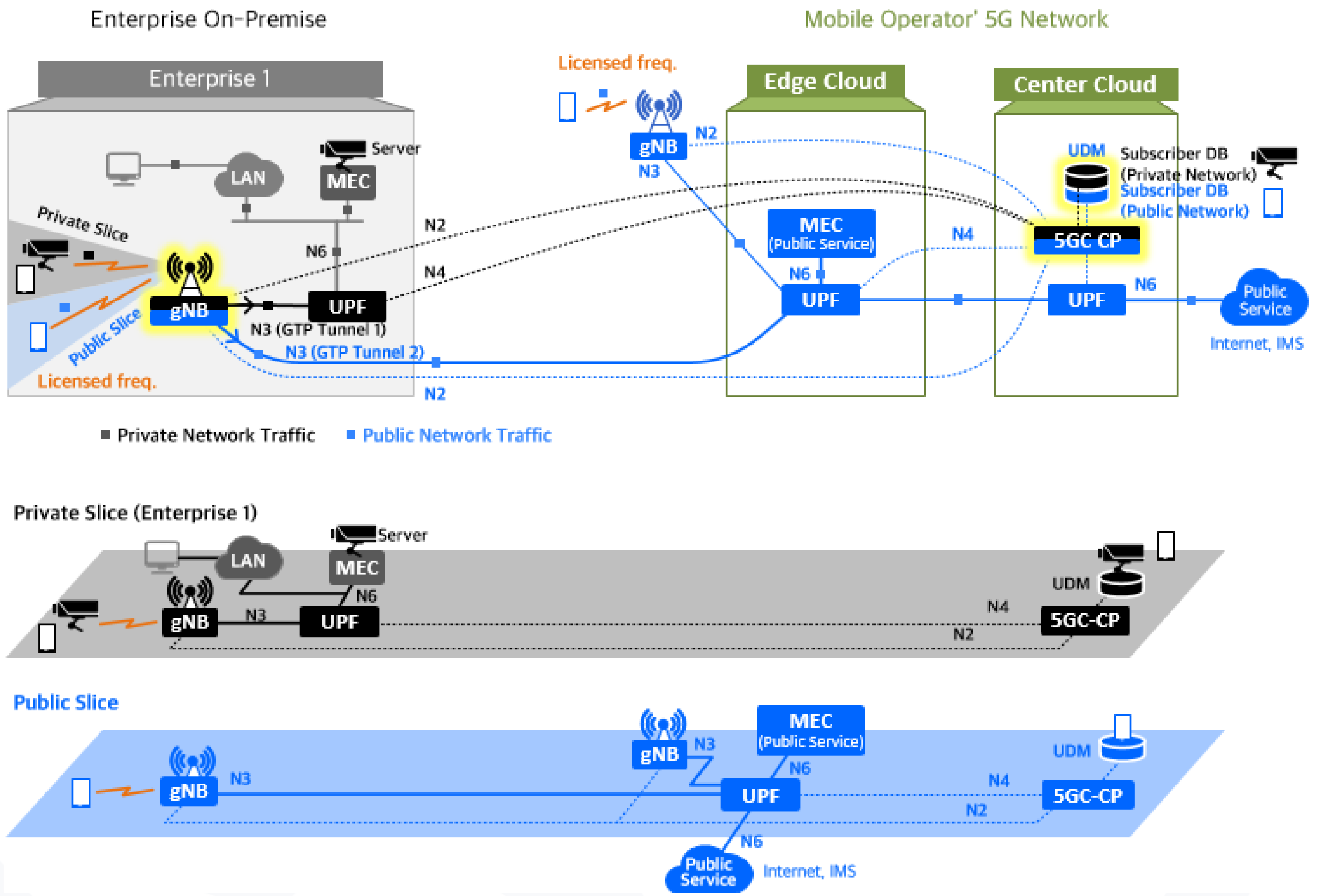
Private 5G vs Public 5G: Isolated Scenario



Source: Netmanias and HFR, Inc

- Pros
  - Security
  - Low Latency
  - No need for backhaul
  - Robust to Mobile Operator faults
- Con
  - High deployment cost
  - High operational personnel cost

Private 5G vs Public 5G: Radio Network and Control Plane Sharing



Source: Netmanias and HFR, Inc

Pros

- Security
- Low Latency
- Lower deployment cost
- Lower operational personnel cost

Con

- Requires backhaul
- Fragile to Mobile Operator faults

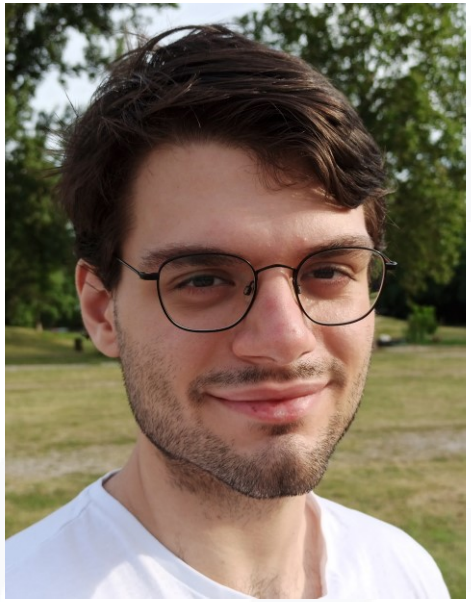
POLITECNICO
MILANO 1863

AI SPRINT
AI IN SECURE PRIVACY-PRESERVING
COMPUTING CONTINUUM

AI-SPRINT project – design phase

AI-SPRINT project has received funding from the European Union Horizoon 2020 research and innovation programme under Grant Agreement **No. 101016577.**

Collaborators



Luca Giacometti

Research assistant @ Politecnico di Milano

luca.giacometti@polimi.it

Prof. Giacomo Verticale

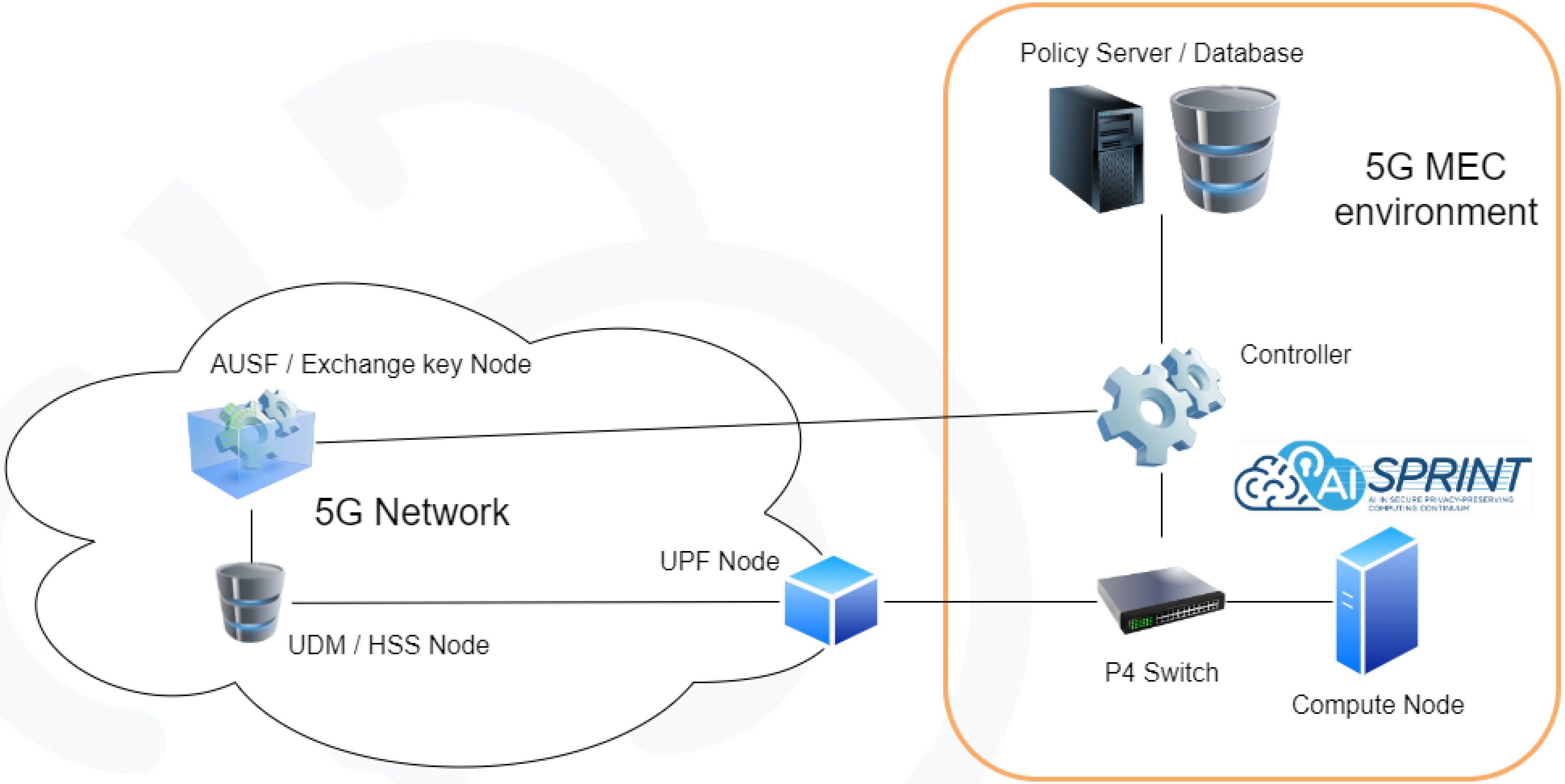
Associate professor @ Politecnico di Milano

giacomo.verticale@polimi.it

Contents

- Scenario - 5G MEC Architecture
- Security Model & Goal
- Authorization Protocol

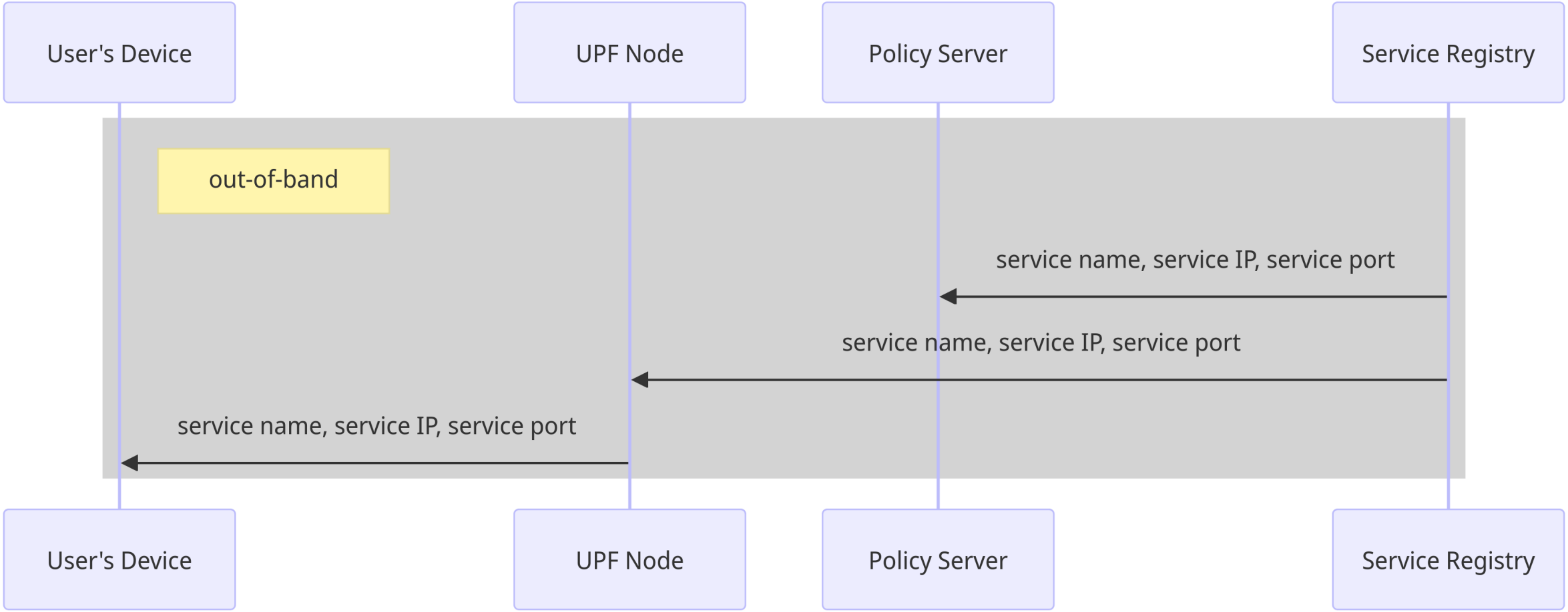
Scenario - 5G MEC Architecture



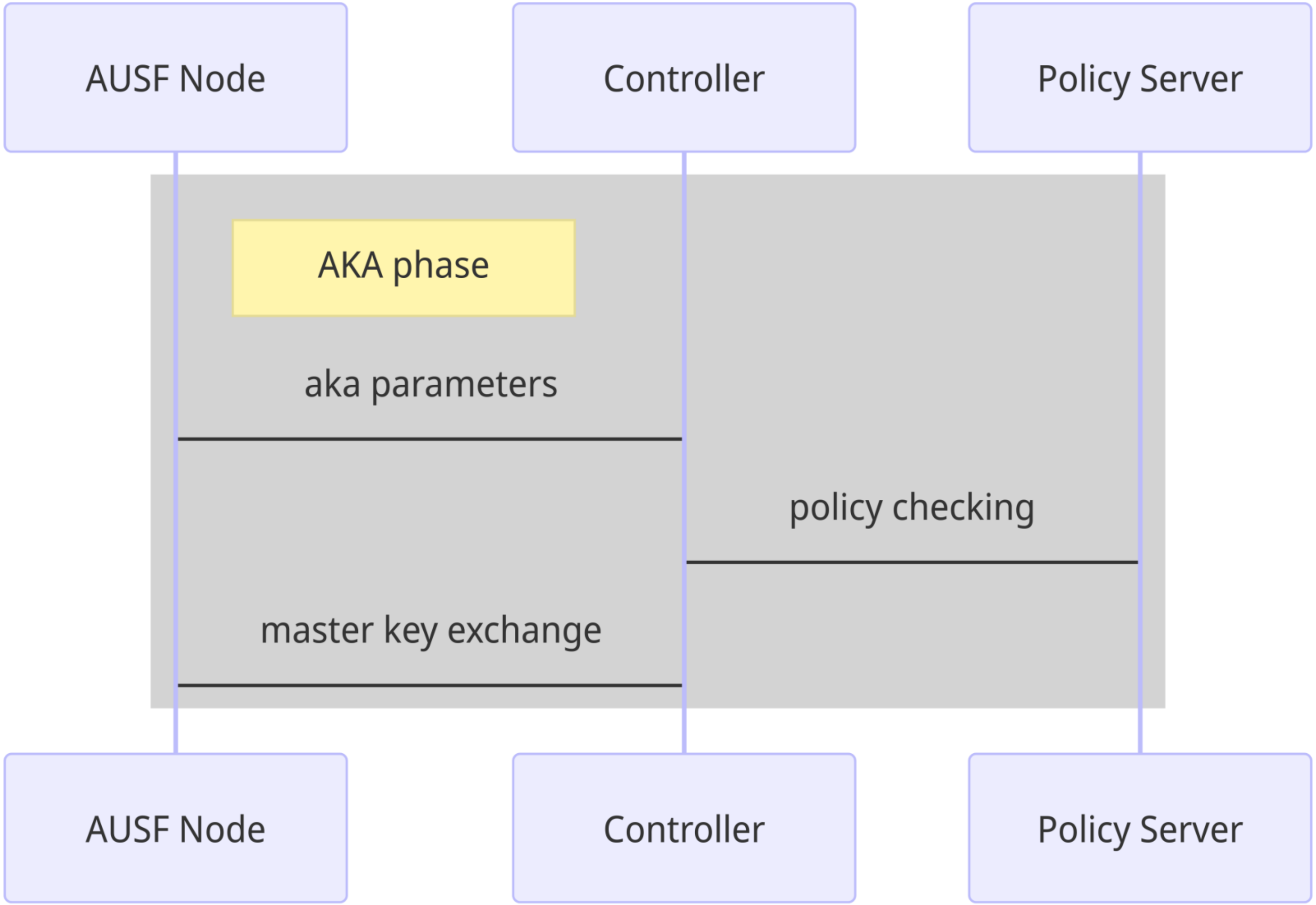
- Services: FaaS paradigm (micro segmentation)
- Project: ZTS principle (white list only)
- Attackers:
  - A compromised MEC node inside the network
  - A machine used to intercept packets

IDEA: To extend 5G device authentication to the MEC node

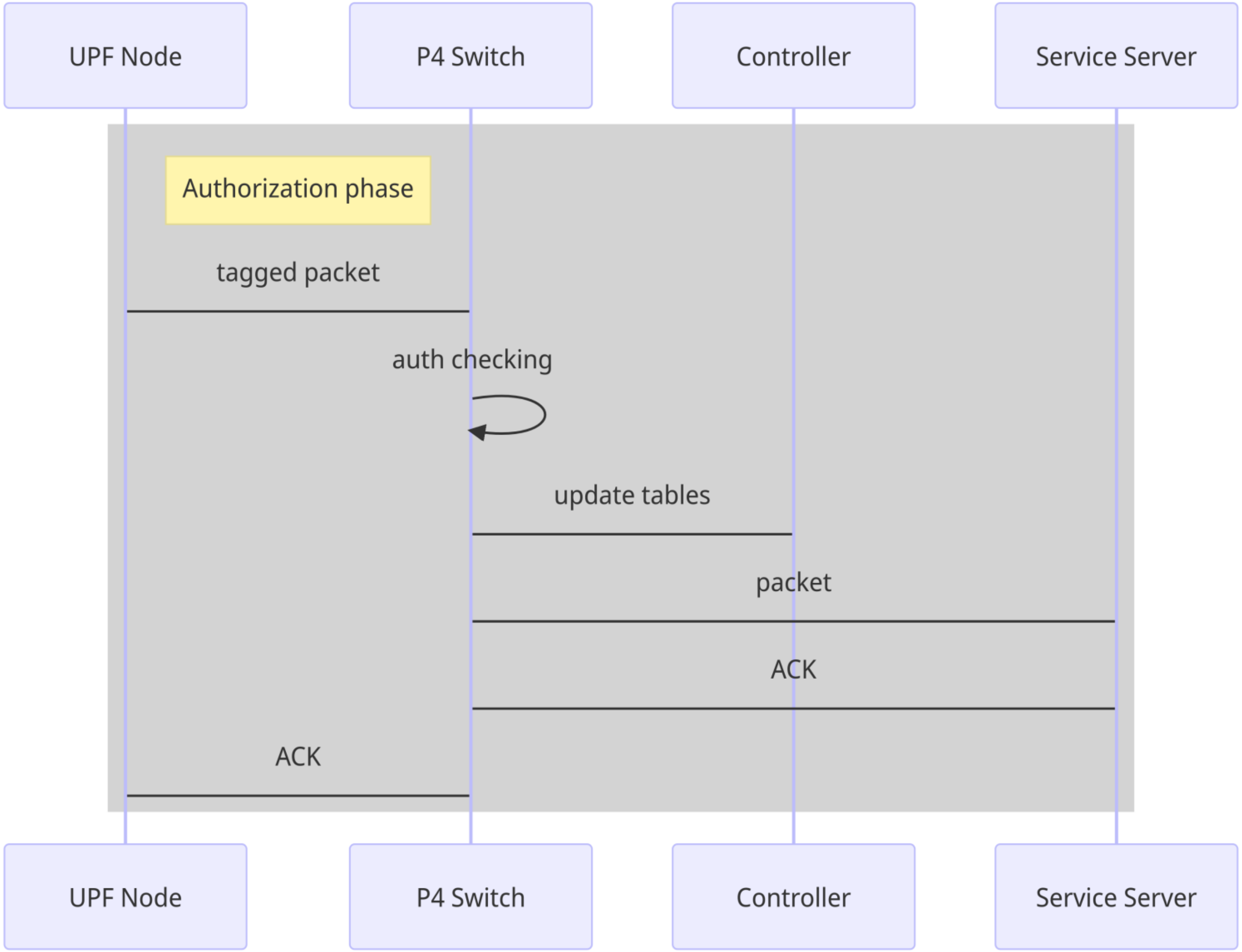
Authorization protocol: Discovery phase



Authorization protocol: AKA phase



Authorization protocol: Authorization phase



- Considered
  - Subversion attack
  - Policy violation
  - Spying and tampering
- NOT Considered
  - Flooding attacks

Implementation part:

- Non Intrusive Load Monitoring (NILM) application
- Usage of SCONE container

Stay Connected

AI SPRINT

AI IN SECURE PRIVACY-PRESERVING COMPUTING CONTINUUM

aisprint-project.eu

@AI_sprint

/AI-SPRINT

The AI-SPRINT project has received funding from the European Union Horizon 2020 research and innovation programme under **Grant Agreement No. 101016577**

POLITECNICO
MILANO 1863

AI SPRINT
AI IN SECURE PRIVACY-PRESERVING
COMPUTING CONTINUUM

AI-SPRINT project – implementation phase

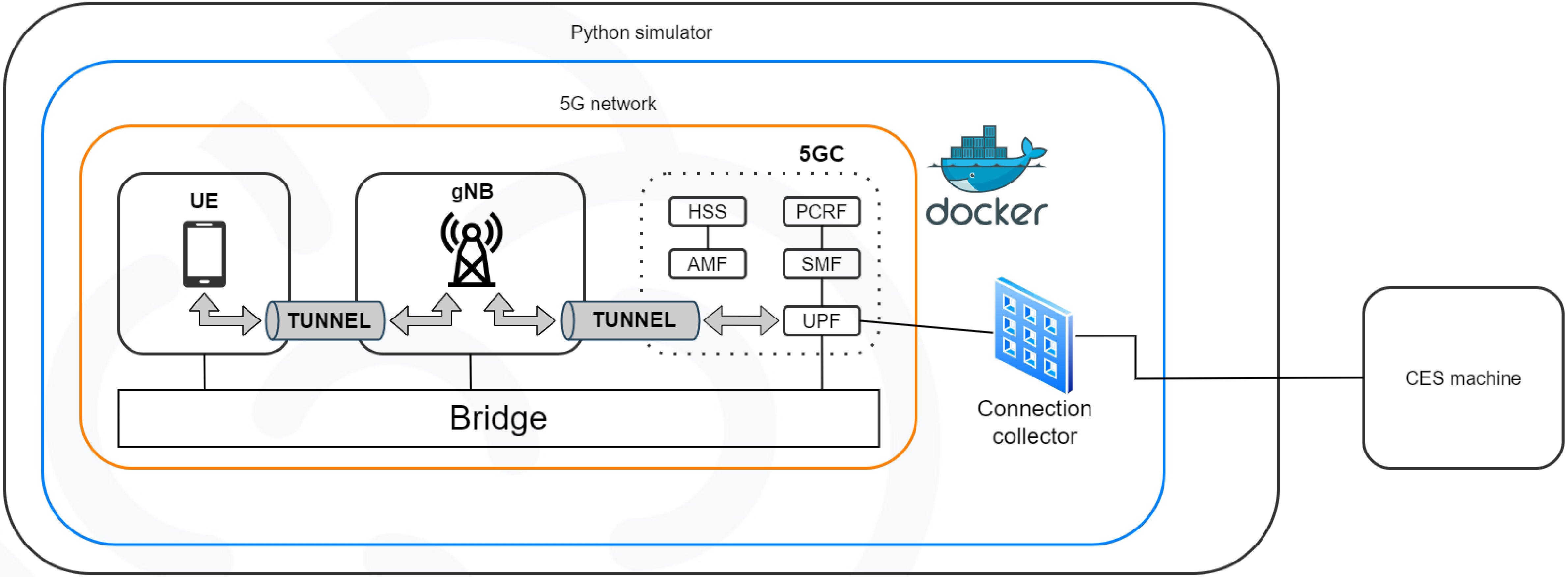
AI-SPRINT project has received funding from the European Union Horizoon 2020 research and innovation programme under Grant Agreement **No. 101016577.**

Contents

- 5G network
- CES component
- MEC server
- Environment integration

Setup

5G network



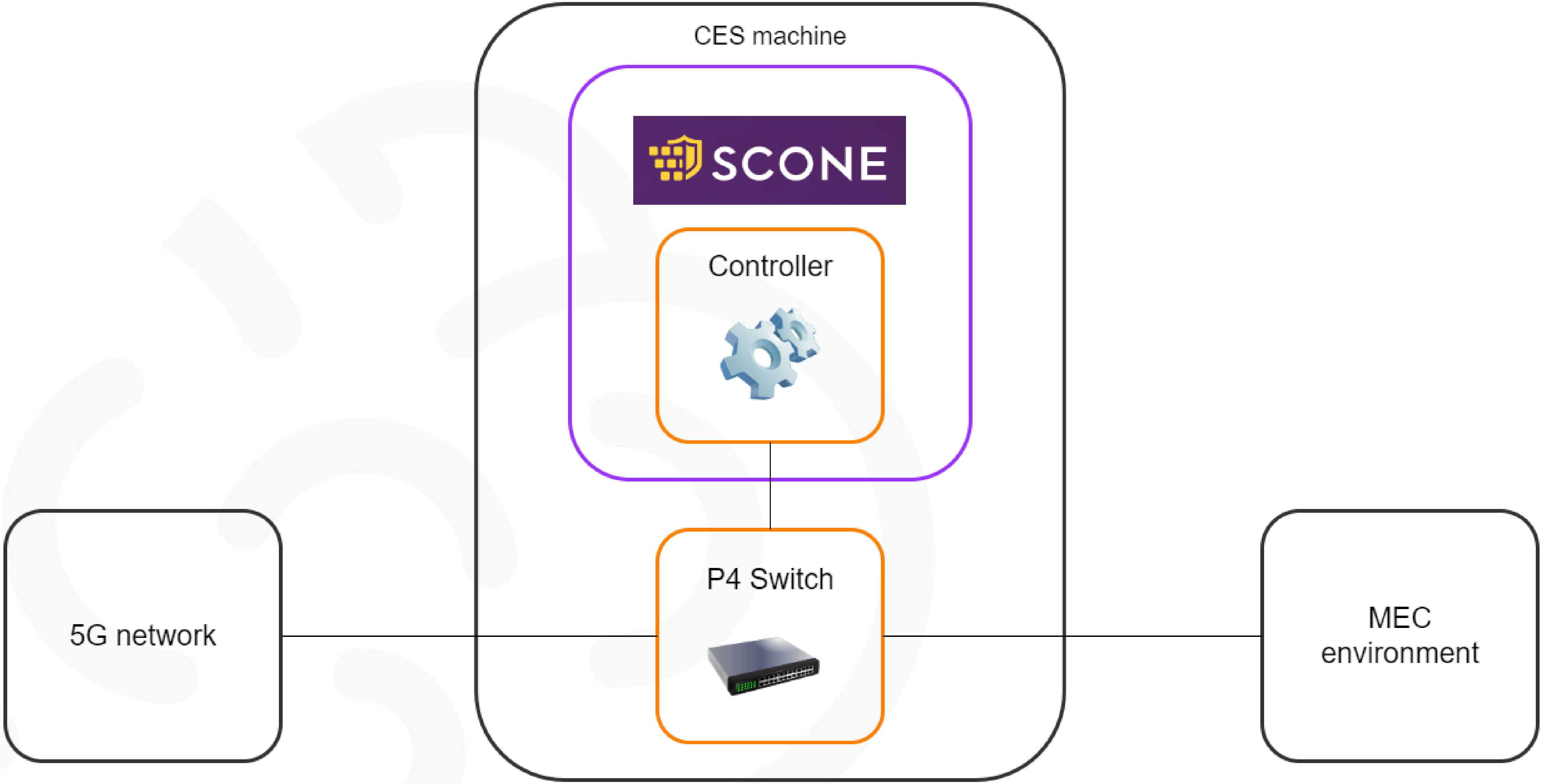
Python simulator
5G network
5GC
UE
gNB
HSS
PCRF
AMF
SMF
UPF
TUNNEL
TUNNEL
docker
Bridge
Connection collector
CES machine

Scone is a confidential computing platform service which allows to run a program hosted remotly in a secure way **trusting only the CPU of the host machine**.

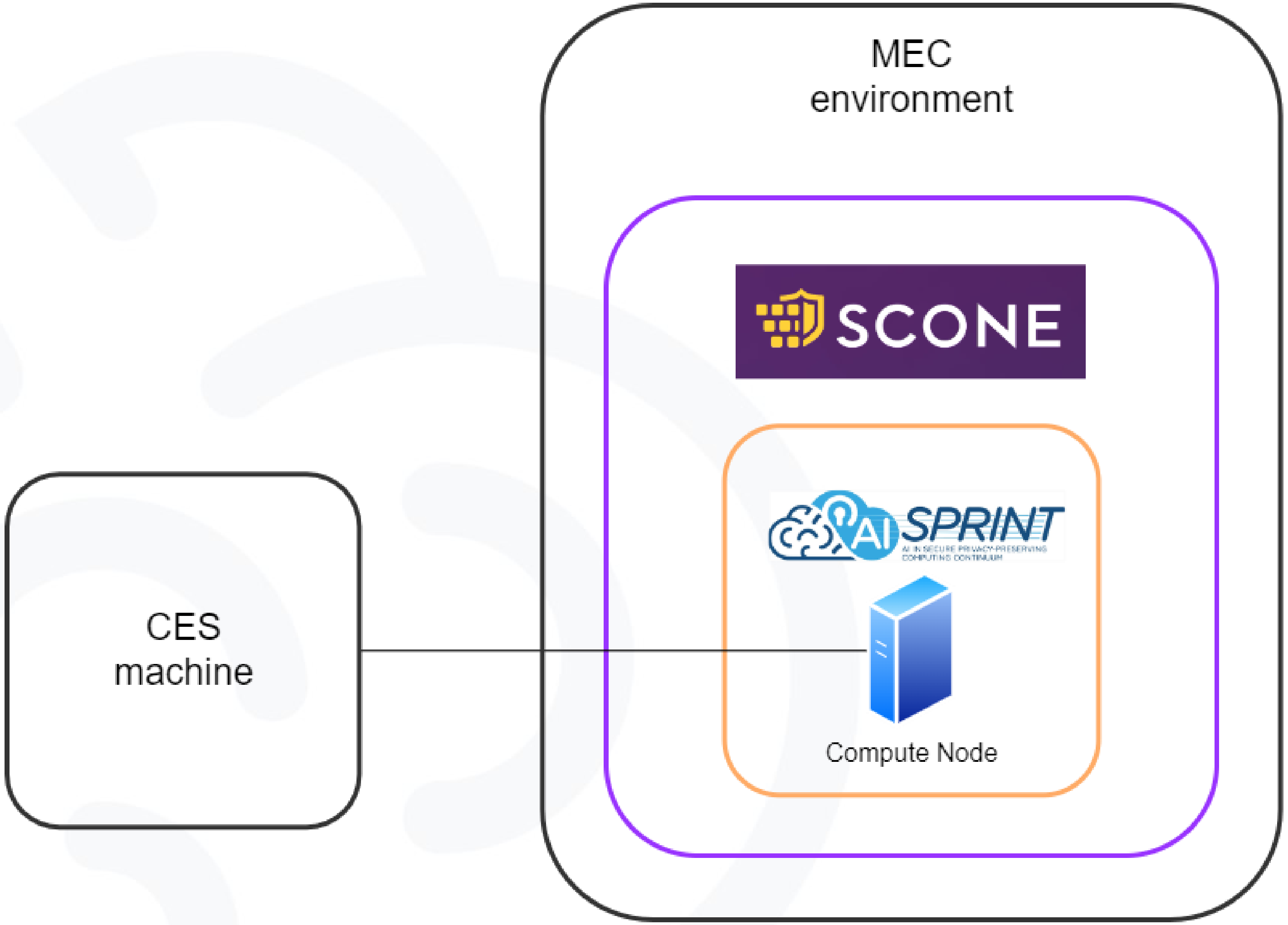
We used it for the AI-SPRINT project:

- Simulation mode (no CAS component)
- Docker standalone deployment

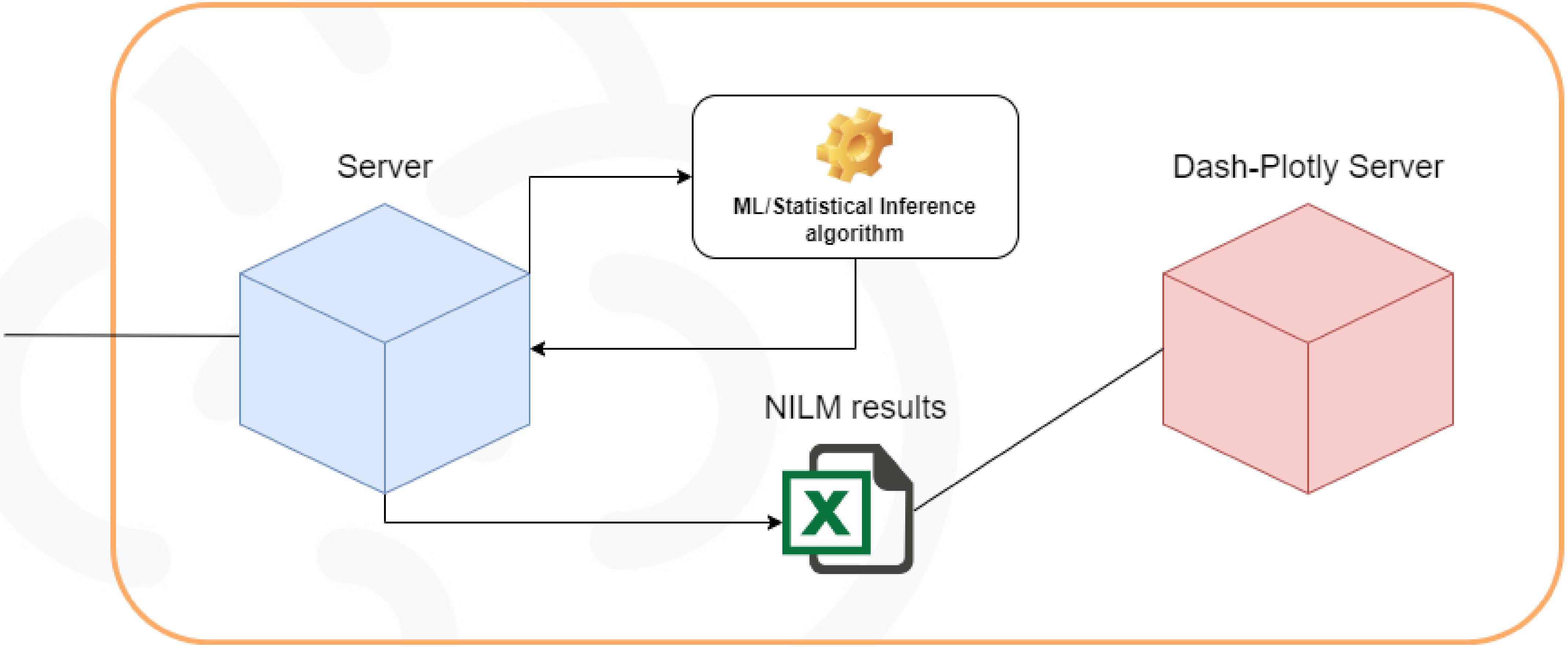
Customer Edge Switch machine



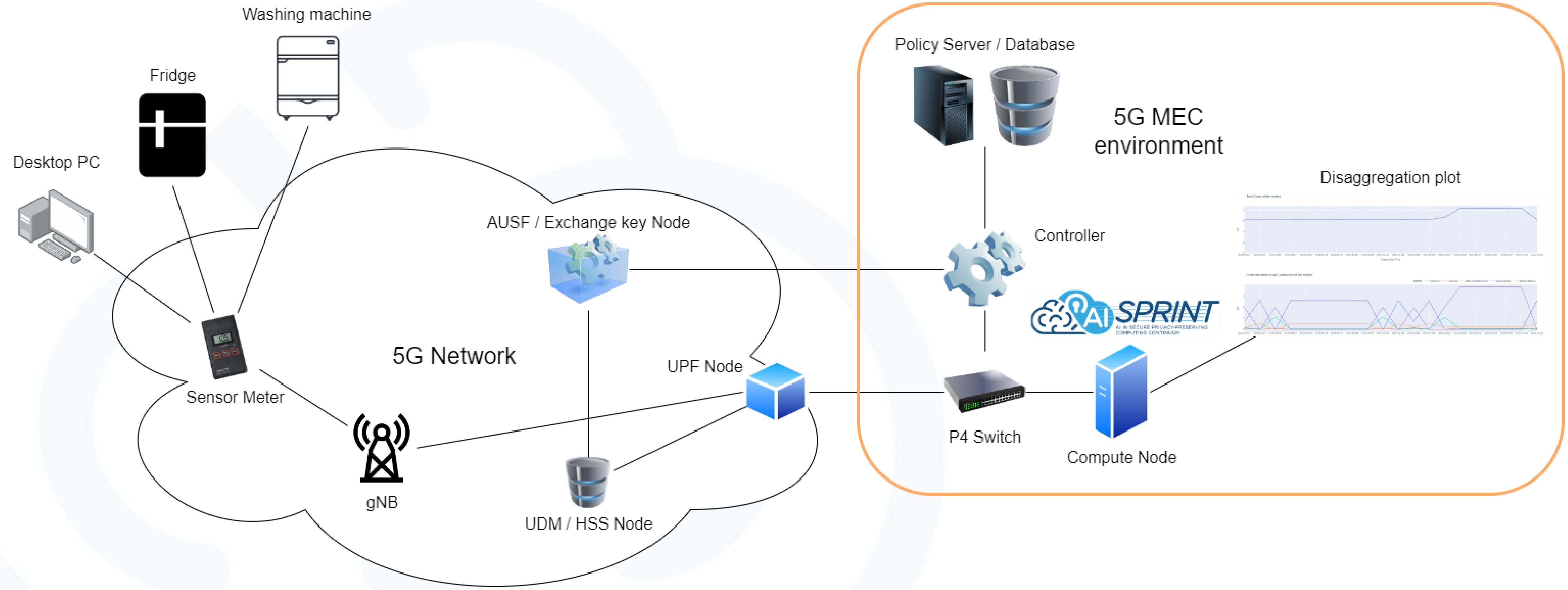
Mobile Edge Computing machine



MEC machine - Compute node



Final Architecture



Stay Connected

AI SPRINT

AI IN SECURE PRIVACY-PRESERVING COMPUTING CONTINUUM

aisprint-project.eu

@AI_sprint

/AI-SPRINT

The AI-SPRINT project has received funding from the European Union Horizon 2020 research and innovation programme under **Grant Agreement No. 101016577**